PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SOFTWARE QUE DETERMINE LA SOLUCIÓN AL PROBLEMA DEL FLUJO MÁXIMO APLICANDO EL ALGORITMO DE FORD-FULKERSON

Tesis para optar por el Título de Ingeniero Informático, que presenta el bachiller:

Jorge Víctor Arangoitia Fernández Baca

ASESOR: Maynard Kong Wong

Lima, mayo del 2012

RESUMEN

El presente proyecto de fin carrera esboza una solución informática al problema del flujo máximo, para lo cual se ha optado por utilizar el algoritmo de Ford-Fulkerson, al ser este el más conocido y difundido, y que permite llegar a una solución exacta del problema en un tiempo relativamente corto. Dicho problema tiene una amplia gama de aplicaciones, que van desde cálculo de rutas disjuntas para redes de comunicaciones, circulación con capacidad, programación de líneas aéreas, selección de proyectos, entre otras.

El problema del flujo máximo fundamentalmente consiste en: dado una red (o grafo) de arcos y nodos, cada arco con una capacidad determinada, y con un nodo fuente y otro sumidero, se trata de hallar la cantidad máxima de material (flujo) que puede circular desde el nodo fuente hasta el nodo sumidero, de manera que el flujo individual que va por cada arco no supere la capacidad de dicho arco; esto último es conocido como restricción de capacidad del arco. Como se verá en la memoria descriptiva, este problema se reduce a uno de investigación de operaciones, es decir, un problema de maximización de una expresión dependiente de una serie de variables, las cuales están sujetas a un conjunto de restricciones.

El algoritmo elegido para la implementación de la solución es el de Ford-Fulkerson, el cual fue propuesto en 1956 en un artículo científico por los matemáticos estadounidenses Lester Randolph Ford Jr. y Delbert Ray Fulkerson, quienes establecieron y demostraron el teorema del flujo máximo - corte mínimo, fundamental para la justificación del algoritmo como proveedor de la solución.

Como se dijo en el párrafo inicial del resumen, existe una vasta y variada cantidad de contextos que pueden modelarse como un problema de flujo máximo, las principales serán brevemente explicadas en la memoria descriptiva, y se deja como trabajo futuro la particularización de esta solución a alguna de las mencionadas situaciones.

FACULTAD DE
CIENCIAS E
INGENIERÍA
ESPECIALIDAD DE
INGENIERÍA INFORMÁTICA

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO:	**ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SOFTWARE QUE DETERMINE LA SOLUCIÓN AL PROBLEMA DEL FLUJO MÁXIMO APLICANDO EL ALGORITMO DE FORD-FULKERSON**
ÁREA:	DESARROLLO DE APLICACIONES
PROPONENTE:	MAYNARD KONG WONG
ASESOR:	MAYNARD KONG WONG
ALUMNO:	JORGE VÍCTOR ARANGOITIA FERNÁNDEZ BACA
CÓDIGO:	20040500
TEMA N°:	_____
FECHA:	16/10/2012

DESCRIPCIÓN

La investigación de operaciones es una rama de las matemáticas que pretende utilizar modelos matemáticos y estadísticos así como algoritmos para estudiar situaciones donde se tiene un problema determinado sujeto a un conjunto de restricciones, y se busca encontrar la solución más óptima para a dicho problema, para así contribuir a la toma de decisiones. Entiéndase, en este contexto, que por optimizar se hace referencia a obtener un valor máximo o mínimo de dicho problema que satisfaga todas las restricciones establecidas. Matemáticamente, los problemas de optimización están conformados por una función matemática o función objetivo, que representa al problema que se desea resolver, y un conjunto de funciones de igualdad o desigualdad, que representan a cada una de las restricciones a las cuales está sometido el mencionado problema. Todas estas funciones son de la forma R^n --> R, es decir, son resultado de operaciones algebraicas entre varias variables. La función objetivo tiene un conjunto de soluciones factibles, cada uno de los cuales otorga valores determinados a cada una de las variables de dicha función que, además de otorgarle un valor a la función objetivo, cumplen con todas las restricciones a las que está sujeta la misma. Lo que busca la investigación de operaciones es encontrar la solución factible que otorgue el valor máximo o mínimo a la función objetivo. De esta manera, al obtener la solución óptima al problema planteado, su aplicación en entornos reales puede generar ganancias o ahorros significativos tanto en tiempo como en dinero.

Uno de los modelos más importantes estudiados por la investigación de operaciones es el problema del flujo máximo, en el cual se tiene un grafo o red compuesto por un conjunto de nodos (en el cual se identifica el o los nodos fuente, el o los nodos sumideros y los nodos intermedios entre la o las fuentes y el o los sumideros) y un conjunto de arcos que conectan los nodos entre sí, cada arco con una capacidad máxima de material que puede transportar. Dado lo anterior, el problema consiste en calcular la máxima cantidad de material que puede ser transportado de la o las fuentes al o a los sumideros sin violar las restricciones de capacidad de los arcos, es decir,

Av. Universitaria 1801
San Miguel, Lima – Perú

Apartado Postal 1761
Lima 100 – Perú

Teléfono:
(511) 626 2000 Anexo 4801

que ningún arco transporte más material de lo que su capacidad establece; ni las restricciones de conservación de flujo, es decir, que la cantidad de material que salga de cada nodo sea exactamente igual a la cantidad que ingresó al mismo. Este problema aparece en situaciones muy disímiles entre sí, como la programación de aerolíneas, la circulación por vías que poseen capacidad límite superior o el diseño de redes de comunicación de datos (estas y otras aplicaciones son estudiadas en la memoria descriptiva).

El presente trabajo propone una solución automatizada a dicho problema, en el cual se utiliza el algoritmo de Ford-Fulkerson, por ser este uno de los métodos de solución más conocidos y difundidos alrededor del mundo. La principal finalidad de este software es que pueda ser utilizado para la enseñanza de la investigación de operaciones, en particular del problema del flujo máximo antes mencionado, sin embargo, el software puede ser utilizado también en situaciones del mundo real que puedan ser modelados a través de dicho problema.

OBJETIVO GENERAL

Realizar el análisis, diseño e implementación de un software que permita obtener la solución automatizada al problema del flujo máximo, para lo cual se hará uso del algoritmo de Ford-Fulkerson.

OBJETIVOS ESPECÍFICOS

A. Identificar los requisitos funcionales y no funcionales del software y realizar el análisis de los mismos.
B. Establecer los módulos que tendrá el software así como la interacción entre ellos.
C. Establecer las estructuras de datos a utilizar para la implementación del algoritmo.
D. Implementar el algoritmo de Ford-Fulkerson, y explicar los principales conceptos y teoremas en los cuales se basa.
E. Realizar la construcción y pruebas del software según el análisis y diseño previamente elaborados.

ALCANCE

El software permitirá calcular de manera automática el flujo máximo en una red de arcos y nodos, cada arco con una capacidad máxima determinada.

Asimismo, trabajará con valores enteros de capacidades y flujos, pues los datos numéricos de dichas variables son fundamentalmente de ese tipo en la mayoría de sus aplicaciones.

El objetivo del proyecto se centra principalmente en el estudio del problema del flujo máximo, así como el análisis e implementación del algoritmo de Ford-Fulkerson para la obtención de la solución de dicho problema, por ello las aplicaciones particulares del problema en contextos específicos serán tocados con menor profundidad, y se dejarán como investigaciones en trabajos futuros. Asimismo, otros temas importantes de investigación de operaciones como son el problema de transporte, la programación de

tareas, la ruta crítica y otros también serán mencionados para implementarse como trabajos futuros.

INDICE

Máximo: 100 páginas

DEDICATORIA

A Dios Padre Creador y Motor de todo lo que existió, existe y existirá.
A mis padres y hermanos que me han soportado tantos años
A mis tíos y primos que siempre han estado ahí pendientes de mí y de mi familia
A mis amigos, constantes compañeros en la carrera de la vida.
A todos aquellos apasionados por las ciencias que contribuyen a formar un mundo mejor.

AGRADECIMIENTOS

A Dios, fuente de toda inspiración y causa fundamental de todo lo que existe
A mis padres y hermanos, cuyo incondicional apoyo fue fundamental para el logro de este proyecto.
A mis tíos y primos, siempre presentes en todo momento.
A mis amigos, con quienes he reído y compartido mil y un aventuras.

TABLA DE CONTENIDOS

ÍNDICE DE IMÁGENES

ÍNDICE DE ECUACIONES

Introducción

El presente proyecto de fin de carrera comprende el desarrollo de un software que permite obtener la solución al problema del flujo máximo, el cual forma parte del estudio de los modelos de redes, tema perteneciente al área de investigación de operaciones. Dicho software, constituye una herramienta que puede ser utilizada tanto en la enseñanza de dicha materia como en aplicaciones prácticas del mencionado modelo en situaciones de la vida real.

Los problemas estudiados por la investigación de operaciones tratan, básicamente, de obtener una solución óptima a una situación planteada, sujeta a un determinado número de restricciones. Uno de los temas más importantes e interesantes de dicha ciencia es el modelo de redes, el cual hace uso de arcos y nodos para la representación gráfica y mejor comprensión de la situación objeto de estudio, uno de los cuales es el ya mencionado problema del flujo máximo.

El problema del flujo máximo consiste en calcular la cantidad máxima de flujo que puede ser transportada de un punto de partida o fuente a uno de llegada o sumidero (1). Entiéndase por flujo, una cantidad medible de materia que puede circular por cualquiera de los arcos. Dicho flujo está sometido a las restricciones de capacidad de cada arco, es decir, a la cantidad máxima de flujo que puede soportar cada uno de los mismos. Ahora bien, el material que fluye a través de la red dependerá de cada situación particular donde el problema se aplique, por ejemplo, el flujo será un líquido en modelos de tuberías, aviones en modelos de programación de aerolíneas, etc. Como en el presente proyecto se estudia el modelo general del problema del flujo máximo, se mencionará el concepto de “flujo” en términos generales.

Como se mencionó anteriormente, el objetivo del proyecto es implementar un software que permita resolver automáticamente el problema del flujo máximo, el cual pueda ser utilizado para la enseñanza de la investigación de operaciones, así como en aplicaciones prácticas que se basen en dicho problema. Para la implementación se ha seguido el modelo en cascada o ciclo de vida clásico del software (2), y para la redacción del documento se ha seguido los documentos de proyectos de tesis en Ingeniería Informática (3), la guía de elaboración del documento de tesis (4) y los

índices base (5), los cuales fueron redactados por los docentes de la Pontificia Universidad Católica del Perú, de la facultad de Ciencias e Ingeniería, especialidad de Ingeniería Informática, y que están publicadas en Internet.

Así, la memoria descriptiva se ha organizado en cinco capítulos, los cuales se explican a continuación:

1. Capítulo 1: Generalidades: Capítulo inicial en el cual se establece la definición del problema que se va a resolver con el presente proyecto, la explicación de la teoría fundamental necesaria para entender la situación estudiada, el estado del arte actual del problema, la planificación del proyecto y la descripción de la solución elaborada.
2. Capítulo 2: Análisis: Se detalla la metodología utilizada para el desarrollo de la solución, los requerimientos, y el análisis de la solución.
3. Capítulo 3: Diseño: En este capítulo se describe la arquitectura de la solución, el diseño de la interfaz gráfica y la arquitectura de la información del software.
4. Capítulo 4: Construcción: Capítulo destinado a sintetizar los principales puntos de la implementación del software y explicar la estrategia de pruebas que se está utilizando.
5. Capítulo 5: Observaciones, conclusiones y recomendaciones: Se mencionan los puntos más resaltantes del proyecto, los resultados obtenidos del mismo y algunos consejos sobre el uso y ampliación del producto desarrollado.

Finalmente, se incluye en el documento los anexos que sustentan el desarrollo del producto y que corresponden a los artefactos elaborados según la metodología seguida.

Capítulo 1: Generalidades

A continuación, se detalla el problema que se pretende resolver con el proyecto, los conceptos necesarios para la comprensión del mismo, su planificación en tiempo, el estado del arte del problema abarcado y finalmente se describe y sustenta la solución propuesta.

1.1. Definición del problema

Previamente a la definición del problema del flujo máximo, es importante puntualizar el concepto de red o grafo, el cual es un conjunto de varios nodos (puntos en el espacio) y arcos (líneas) que salen de un determinado nodo y van hacia otro. Los grafos pueden ser dirigidos, cuando cada arco tiene un nodo origen y uno destino (es decir los arcos tienen un sentido), o no dirigidos en caso contrario. Las figuras 1.2 y 1.3 ilustran de modo gráfico los conceptos de grafo dirigido y no dirigido, asimismo, más adelante en el presente capítulo se establece la definición formal y matemática de los grafos.

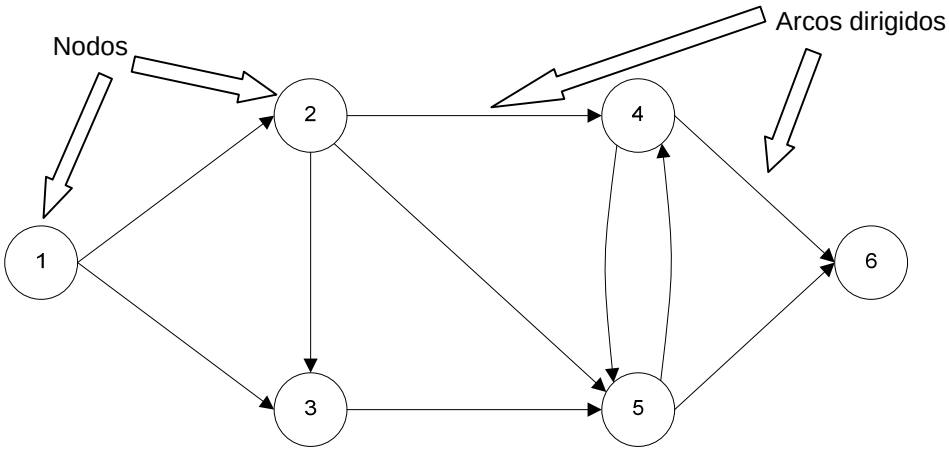

Fig. 1. 1: Red o grafo dirigido

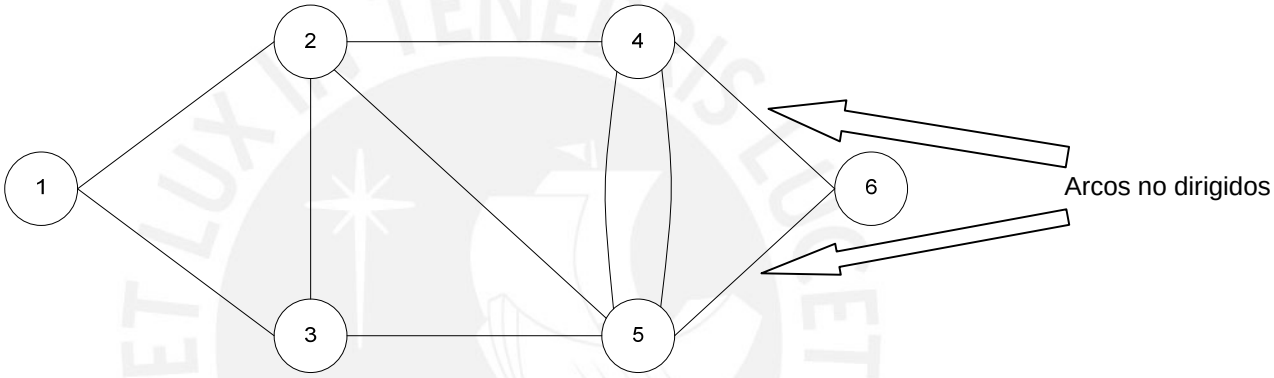

Fig. 1. 2: Red o grafo no dirigido

Dicho lo anterior, el problema del flujo máximo en su forma más pura consiste en que: Existe un grafo dirigido o no dirigido (comúnmente dirigido en la mayoría de aplicaciones reales), donde uno de los vértices es considerado como la "fuente" y otro como el "sumidero", de tal manera que algún material u objeto puede fluir desde la fuente hasta el sumidero; a la cantidad de material u objeto que circula por el grafo se le denomina flujo. Entre la fuente y el sumidero existe una cantidad determinada de nodos interconectados entre sí a través de arcos, cada uno de estos arcos tiene una capacidad máxima que puede transportar entre los nodos que conecta, la cual puede variar de arco a arco. Esto quiere decir, que cada arco solo podrá soportar un flujo menor o igual a su capacidad, de tal manera que si un flujo mayor quiere discurrir a través de un arco, solo una parte de dicho flujo (de valor igual a la capacidad de ese arco) viajará a través de él, y el resto deberá ir por otro arco que salga del mismo nodo, de no haber otro arco, entonces el flujo se verá reducido.

El objetivo del problema del flujo máximo es determinar la máxima cantidad de material u objetos (flujo) que pueden fluir en el grafo desde la fuente hacia el sumidero. En aplicaciones del mundo real, conocer el valor del flujo máximo permite a la fuente saber exactamente cuánto producir y enviar a través de una ruta sin generar desperdicios.

La forma más básica en que este problema se encarna sería una compañía manufacturera que elabora un producto en su fábrica y lo envía a su centro de venta en una diferente ciudad del país. Además existen dos paradas entre ambos puntos para los camiones que trasladan los productos. Un camión con una cantidad máxima de productos sale de la fábrica a la primera parada, descarga y luego regresa a la fábrica; de la primera parada sale otro camión con productos hacia la segunda parada, descarga y regresa; y así similarmente para la segunda parada hacia el centro de ventas. Si todos los camiones tienen la misma capacidad de transporte de productos, entonces no habría ningún problema, pues la cantidad a transportar será constante; sin embargo, si la capacidad de cada camión es distinta, en las paradas puede generarse un remanente, por lo que la compañía incurrirá en costos de almacenamiento y conservación de dicho remanente. El modelo del flujo máximo permite, en este caso, calcular la cantidad máxima a producir por la fábrica, a fin de minimizar los costos de almacenaje y conservación (7).

La solución a la situación antes mencionada es simple: el valor óptimo es la capacidad del camión más pequeño. En términos del problema del flujo máximo, los puntos de parada serán los nodos, el camino entre ellos los arcos, y la capacidad del arco la máxima cantidad que puede transportar cada camión. En consecuencia, para este caso particular, el flujo máximo es la capacidad de arco mínima. Asimismo, de agregarse más nodos al grafo que son accedidos de forma secuencial, es decir, uno tras otro, la solución sigue siendo la misma que en el caso anterior. La figura 1.3 ilustra este hecho.

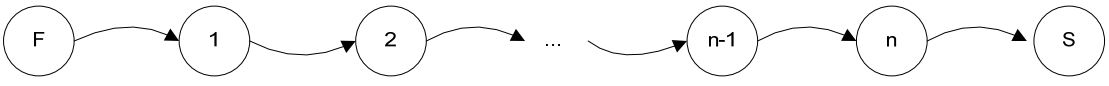

Fig. 1. 3: Red con nodos accedidos secuencialmente.

Sin embargo, la resolución del problema de flujo máximo se hace más compleja cuando la red presenta nodos que son accedidos de forma no secuencial es decir, cuando a partir de un nodo cualquiera se puede acceder a varios otros nodos (como la red de la 1.1), asimismo, a mayor cantidad de nodos y arcos, aún mayor será la dificultad de encontrar el flujo máximo. Si bien, existen algoritmos que permiten obtener el flujo máximo para estas redes (algunos de ellos se describen en el punto 1.3.1), a medida que aumenta el tamaño del grafo, se va haciendo poco práctico resolver el problema manualmente, pues tomaría un tiempo bastante elevado y una alta exposición al riesgo de error u omisión, riesgo inherente al trabajo manual. En ese sentido se hace necesario el uso un software automatizado para poder encontrar una solución óptima a dicho problema en un tiempo razonablemente corto.

1.2. Marco conceptual

A continuación se menciona la teoría fundamental básica necesaria para el entendimiento del problema del flujo máximo.

1.2.1. Investigación de operaciones

La investigación de operaciones es la parte de la matemática que estudia los problemas acerca de cómo conducir y coordinar las operaciones dentro de una organización, a fin de optimizarlas (8). Básicamente, cualquier problema de investigación de operaciones busca maximizar o minimizar una función matemática (función objetivo) sin transgredir un número determinado de restricciones.

Una solución de un problema de investigación de operaciones es factible si satisface todas las restricciones, y es óptima si, además de ser factible, produce el mejor valor (máximo o mínimo) de la función objetivo (9).

1.2.2. Modelos de redes

Los modelos de redes constituyen una forma de plantear problemas o situaciones reales a través del uso de redes o grafos. Muchos problemas de optimización importantes se analizan mejor por medio de una representación gráfica de red, dicha gráfica facilita la comprensión de dichos problemas así como la obtención de su solución (1).

1.2.2.1. Red o grafo

Una red o grafo es un conjunto de nodos (vértices) y los arcos (aristas) que los conectan. Matemáticamente, un grafo G se define como:

$$G = (V, E)$$

Ecuación 1. 1: Representación matemática de un grafo

Donde

V = Conjunto de vértices del grafo

E = Conjunto de arcos del grafo.

Además se define también:

$|V|$ = Cantidad de vértices del grafo.

$|E|$ = Cantidad de arcos del grafo. (10)

1.2.2.2. Nodo

Los nodos son los puntos o vértices que forman parte de una red o grafo. Generalmente tienen asignado un número o letra que los identifica.

1.2.2.3. Arco

Los arcos son las aristas o líneas que unen los nodos. Estas aristas pueden tener o no una dirección de movimiento entre los nodos que une. Matemáticamente se le representa como un para ordenado, donde sus elementos son los nodos que conecta. La figura 1.4 representa una red simple de dos nodos y un arco.

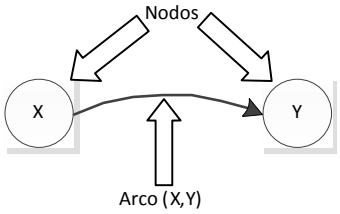

Fig. 1. 4: Red simple

1.2.2.4. Cadena

Secuencia de arcos consecutivos (ver figura 1.5)

Fig. 1. 5: Cadena

1.2.2.5. Trayectoria

Cadena en la que el nodo terminal de cada arco es idéntico al nodo inicial del arco siguiente. La figura 1.5 no es una trayectoria, pues el arco (1,2) termina en el nodo 2, sin embargo el arco siguiente, (3,2), inicia en el nodo 3. La figura 1.6 representa una trayectoria.

Fig. 1. 6: Trayectoria

1.2.3. Problema del flujo máximo

Aunque se le atribuye al norteamericano George Dantzig la definición y modelado del problema del flujo máximo en los inicios de los años 50, existen varios documentos anteriores que constituyen un estudio básico del mismo que datan de 1930. Por ejemplo, el problema de transporte fue ampliamente estudiado en la Unión Soviética. Este problema consiste en obtener el número óptimo de unidades a trasladar desde ciertos puntos de origen hacia ciertos puntos de destino, a fin de minimizar los costos de transporte entre dichos punto, además, se conoce la oferta de unidades de los puntos de origen y la demanda de las mismas en los puntos de destino. El director de la Comisaría Nacional de Transporte de la entonces URSS, el comisario A.N. Tolstoi, estudió la solución óptima para el transporte de carga a través de toda la Unión Soviética vía el sistema de ferrocarriles con múltiples fuentes y varios destinos. En su artículo “Methods of Finding Minimum Total Kilometrage in Cargo-Transportation Planning in Space”, plantea varias maneras de cómo hallar la solución a dicho problema (6). Tolstoi aplicó sus investigaciones a gran escala en las necesidades de

transporte de una vasta cantidad de productos de distintos usos en la Unión Soviética. Aunque el trabajo de Tolstoi no fue exactamente acerca del problema del flujo máximo, fue una de las primeras aproximaciones al mismo.

Si bien existen trabajos similares a los de Tolstoi en los mismos años, se atribuye a George Bernard Dantzig la definición formal del problema del flujo máximo en 1951, y a L. R. Ford y D. R. Fulkerson el primer algoritmo conocido de solución del mismo. Aunque a partir de allí se han desarrollado nuevos algoritmos que disminuyen el tiempo de ejecución del cálculo de la solución al problema, estos básicamente constituyen mejoras al algoritmo de Ford y Fulkerson.

Antes de explicar la naturaleza del problema del flujo máximo, se describirán algunos conceptos necesarios para su comprensión.

1.2.3.1. Fuente

Se denomina fuente al nodo inicial o punto de partida desde el cual se inicia el transporte del flujo.

1.2.3.2. Sumidero

El sumidero es el nodo final o punto terminal hacia el cual debe llegar el flujo.

1.2.3.3. Capacidad del arco

La capacidad de un arco es la máxima cantidad de flujo que dicho arco puede trasladar. Matemáticamente, se define como una función sobre el arco: $c(u, v)$. (10)

1.2.3.4. Red de flujo

Una red de flujo es un grafo dirigido en el cual cada arista tiene capacidad no negativa, y, además, todos sus vértices se comunican con la fuente y con el sumidero a través de una trayectoria.

Matemáticamente, dado un grafo $G = (V, E)$, dicho grafo es una red de flujo si:

$\forall (u, v) \in E : c(u, v) \geq 0$

$\forall v \in V : \exists trayectoria : fuente \rightarrow v \rightarrow sumidero$

$|E| \geq |V| - 1$

(10)

1.2.3.5. Flujo

El flujo se refiere al movimiento del material en una red de flujo. Matemáticamente se le representa como una función $f(u, v)$ o una variable con subíndices $x_{i,j}$.El flujo cumple tres propiedades:

a. Restricción de capacidad: El flujo en un arco tiene valor numérico menor o igual a la capacidad del arco, es decir:

$$f(u, v) \leq c(u, v)$$

Ecuación 1. 2: Restricción de capacidad.

b. Simetría: El flujo en un arco puede representarse como un flujo de valor negativo en sentido inverso en el mismo arco, es decir:

$$f(u, v) = -f(v, u)$$

Ecuación 1. 3: Ecuación de simetría.

c. Conservación de flujo: El flujo total que ingresa a un nodo es igual al flujo total que sale de dicho nodo, es decir:

$$\sum f(X, v) = \sum f(v, Y)$$

Ecuación 1. 4: Conservación de flujo

Donde X es el conjunto de nodos origen de los arcos que tienen como destino a v, e Y es el conjunto de nodos destino de los arcos que tienen como origen a v.(10)

1.2.3.6. Problema del flujo máximo

El objetivo del problema del flujo máximo es calcular la máxima cantidad de un material que puede ser transportado de la fuente al sumidero sin violar las restricciones de capacidad de los arcos (10).

Por ejemplo, se tiene la siguiente red de flujo:

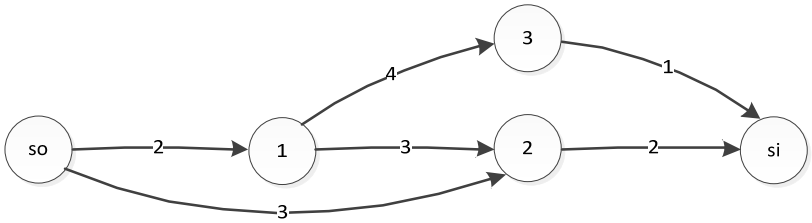

Fig. 1. 7: Ejemplo de red de flujo

Donde:

so = fuente

si = sumidero

Si denominamos $x_{i,j}$ a la cantidad de flujo que va desde el nodo "i" al nodo "j".

Entonces, de acuerdo a las propiedades del flujo, un flujo factible cumplirá que:

a. $0 \leq x_{i,j} \leq c(i,j)$

b. $x_{i,j} = -x_{j,i}$

c. $flujo_{que\,entra\,en\,el\,nodo_i} = flujo_{que\,sale\,del\,nodo_i}$

Si al grafo de la figura 1.7 se le agrega el arco de capacidad x_0 que va de "si" a "so"

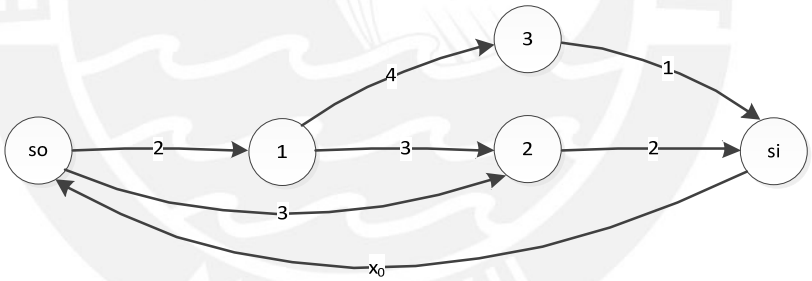

Fig. 1. 8: Red de flujo con arco que va del sumidero a la fuente

Entonces el problema del flujo máximo se convierte en un problema de MAXIMIZACIÓN, que cumple la forma general de todo problema de investigación de operaciones (maximización o minimización de una función sujeta a determinadas restricciones):

$$MAX: z = x_0$$

$$s.a$$

$$x_{i,f} \geq 0$$
$$x_{so1} \leq 2$$
$$x_{so2} \leq 3$$
$$x_{1,2} \leq 3$$
$$x_{1,3} \leq 4$$
$$x_{2,st} \leq 2$$
$$x_{3,st} \leq 1$$
$$x_{1,2} \leq 3$$

Restricciones de capacidad

$$x_0 = x_{so1} + x_{so2}$$
$$x_{so1} = x_{1,2} + x_{1,3}$$
$$x_{1,2} + x_{so2} = x_{2,st}$$
$$x_{1,3} = x_{3,st}$$
$$x_{3,st} + x_{2,st} = x_0$$

Restricciones de conservación de flujo

Ecuación 1. 5: Forma general del problema del flujo máximo

1.2.3.7. Redes con múltiples fuentes y sumideros

En las diferentes aplicaciones del problema del flujo máximo, las redes de flujo generadas no necesariamente poseen una sola fuente y un solo sumidero, sino que pueden tener dos, tres o más de ellas.

Sin embargo, cualquiera sea la cantidad de fuentes o sumideros, esta situación se puede reducir a un problema de flujo máximo ordinario, lo único que se debe hacer es agregar una súper-fuente con arcos de capacidad infinita (o muy grande) que partan de ella y vayan hacia cada una de las fuentes originales de la red. Similarmente, se agrega un súper-sumidero con arcos de capacidad infinita (o muy grande) que partan de cada uno de los sumideros originales de la red y vayan hacia el súper-sumidero creado. Hecho esto, el problema se verá reducido a un problema de flujo máximo conocido (10).

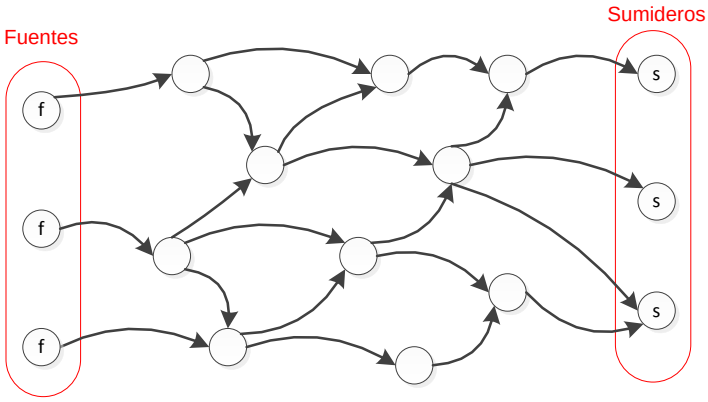

Fig. 1. 9: Red con varias fuentes y varios sumideros

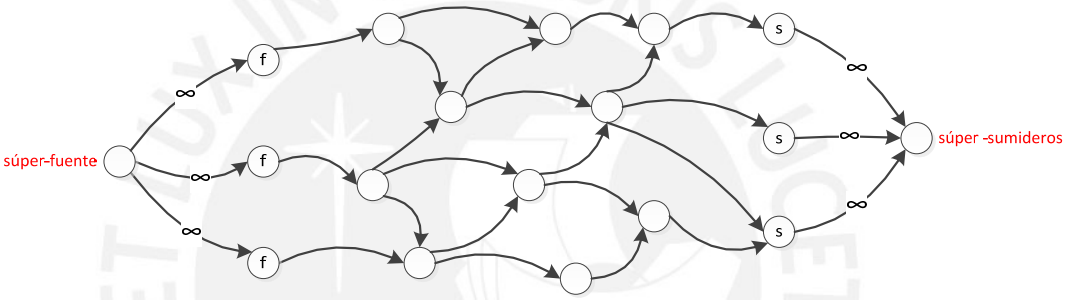

Fig. 1. 10: Red con la súper-fuente y súper-sumidero agregados

1.2.4. Aplicaciones del problema del flujo máximo

El problema del flujo máximo es uno de los más importantes de la investigación de operaciones, pues tiene una gran cantidad de aplicaciones en situaciones reales, algunas de ellas son descritas a continuación (11).

1.2.4.1. Problema de las trayectorias disjuntas o problema de conectividad

Dos trayectorias son disjuntas si no tienen ni un solo arco en común. El problema de las trayectorias disjuntas básicamente consiste en calcular la mayor cantidad de trayectorias disjuntas en una red dada.

Por ejemplo, en la siguiente red:

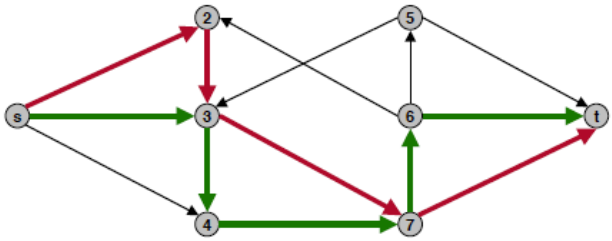

Fig. 1. 11: Ejemplo de red para el problema de las trayectorias disjuntas

Fácilmente se puede observar que, como máximo, existen 2 trayectorias disjuntas.
Este problema, se puede modelar como uno de flujo máximo, basta otorgar una capacidad de uno a todos los arcos; así, el flujo máximo calculado será la cantidad máxima de trayectorias disjuntas que existan en la red. Este hecho se gracias a que, como cada arco tiene capacidad de uno, para que el flujo se incremente en una unidad deberá existir una trayectoria nueva que incluya arcos por los cuales aún no circule flujo.

La aplicación principal de este problema es la conectividad de redes de comunicación, pues para conectar dos puntos extremos de la red es necesaria una trayectoria disjunta, por ende, el máximo número de conexiones que se pueden hacer en una red de comunicación es numéricamente igual al máximo número de trayectorias disjuntas existentes en dicha red.

1.2.4.2. Emparejamiento bipartito de máxima cardinalidad.

Dado un grafo G, un emparejamiento M es un subconjunto de G, tal que cada nodo de G esté en, a lo más, un arco de M.
Gráficamente, en la figura 1.12 se pueden observar los arcos del emparejamiento resaltados en la siguiente red (G), arcos que a su vez constituyen la red M:

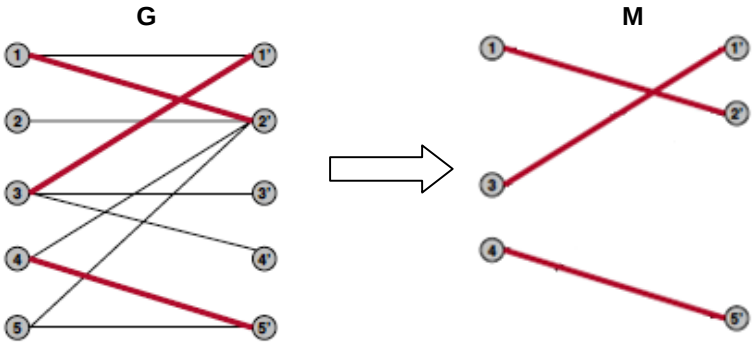

Fig. 1. 12: Red con un emparejamiento bipartito y red resultante

El problema de emparejamiento bipartito de máxima cardinalidad consiste en calcular la mayor cantidad de emparejamientos que se pueden encontrar en una red dada. Este problema se modela como uno de flujo máximo, pues se considera a los nodos de la izquierda como fuentes, a los de la derecha como sumideros, y los arcos como si tuvieran capacidad de valor numérico igual a uno (porque cada nodo solo puede "emparejarse" con un y solo un nodo). Por esta razón, el problema se transforma a uno de flujo máximo de varias fuentes y varios sumideros, el cual, según lo mencionado en el punto 1.2.3.7, se resuelve agregando una súper-fuente y un súper-sumidero. Así:

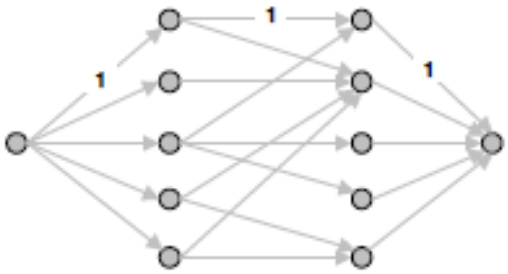

Fig. 1. 13: Red con emparejamiento planteado como un problema de flujo máximo

Las aplicaciones de este modelo pueden encontrar en situaciones de emparejamiento, ya sea de parejas de baile, actividad de casamentero (1), etc.

1.2.4.3. Circulación por demanda

Dado un grafo dirigido $G = (V, E)$, con arcos con capacidades $c(u, v)$, y nodos que ofrecen suministros o solicitan demandas $d(v)$, se define circulación a una función f que satisface las siguientes propiedades:

1. Capacidad: $\forall e \in E: 0 \leq f(e) \leq c(e)$
2. Conservación: $\sum_{e_{entran_{a_v}}} f(e) - \sum_{e_{salen_{de_v}}} f(e) = d(v)$

El problema de circulación por demanda consiste en calcular una circulación existente en el grafo G antes dado. Gráficamente, en cada nodo se coloca el valor del suministro o demanda que representa, colocando los suministros como valores negativos y las demandas como positivas, para cumplir la propiedad de conservación antes mencionada. La figura 1.14 ilustra un ejemplo de este problema.

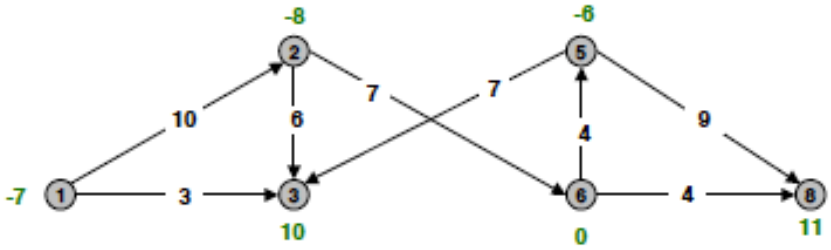

Fig. 1. 14: Red con suministros y demandas

Este problema puede modelarse como uno de flujo máximo, pues se puede crear una súper-fuente conectada hacia todos los nodos suministros, con arcos de capacidades iguales a los valores positivos de los suministros; y un súper-sumidero hacia el cual se conecten todos los nodos demanda a través de arcos cuya capacidad será el valor de las demandas, de manera que este problema toma la forma de uno de flujo máximo (ver figura 1.15)

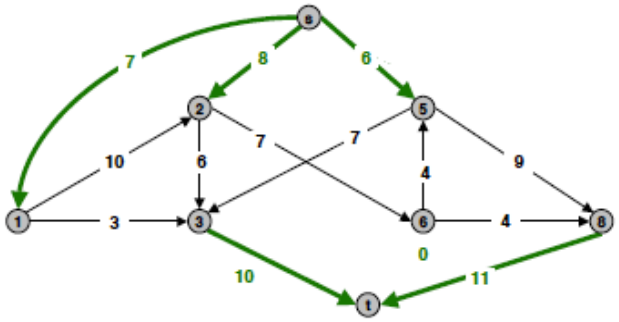

Fig. 1. 15: Circulación por demanda como un problema de flujo máximo

Este problema tiene una amplia gama de aplicaciones en temas de transporte de productos y provisiones, así como sus variantes.

1.2.4.4. Circulación por demanda con límites inferiores

Es una variante del problema de circulación por demanda en el cual, aparte de las capacidades de cada arco, también se agregan valores (l) correspondientes a los límites inferiores a los mismos, de tal manera que se fuerza la existencia de flujo por ciertos arcos; cabe señalar también que las propiedades de capacidad y conservación del problema original se siguen respetando (ver figura 1.16).

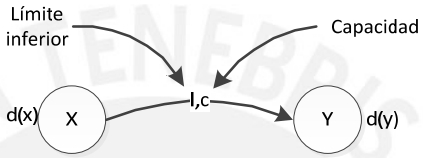

Fig. 1. 16: Arco con capacidad y límite inferior

Para resolver este problema, se le transforma en uno de circulación por demanda simple, para lo cual se crea una nueva red de estructura (nodos y arcos) similar a la del problema original, cuyas arcos tengan capacidades (c') iguales a la capacidad original del arco menos el valor del límite inferior del mismo, y cuyos nodos tengan como demanda (d') el valor de la demanda original más la suma de los valores de los límites inferiores de todos los arcos que entren a dicho nodo, menos la suma de los valores de los límites inferiores de todos los arcos que salen del mismo nodo. Es decir:

$$c'(e) = c(e) - l(e)$$

$$d'(v) = d(v) + \sum_{e_{entran_v}} l(e) - \sum_{e_{salen_v}} l(e)$$

Hecho esto, el problema se convierte en uno de circulación por demanda simple.

1.2.4.5. Navegación de aerolíneas

El objetivo de este problema es determinar la máxima cantidad de vuelos de conexión que se pueden concretar entre dos ciudades, entre las cuales existen ciudades donde los aviones deben realizar escala para luego continuar el viaje. Además, los

aeropuertos de las ciudades tienen un espacio limitado de aterrizaje, lo cual limita la cantidad de aviones que pueden llegar a ellos.

Dado lo anterior, se puede determinar que este problema se puede modelar como uno de flujo máximo, donde los nodos serán las ciudades, la fuente será la ciudad origen, el sumidero la ciudad destino, los arcos las rutas que interconectan las ciudades, y la capacidad de cada arco la cantidad de aviones que pueden aterrizar en la ciudad hacia donde va el arco (1).

1.2.4.6. Selección de proyectos

Se tiene una determinada cantidad de proyectos (p_i) y una determinada cantidad de equipos (q_i). Los proyectos generan un beneficio $b(p_i)$, y el uso de cada equipo conlleva un costo $c(q_i)$. Cada proyecto requiere un número de equipos, y cada equipo puede ser compartido por uno o más proyectos. El problema consiste en determinar qué proyectos deberían ser seleccionados y qué equipos deberían ser comprados para maximizar las ganancias.

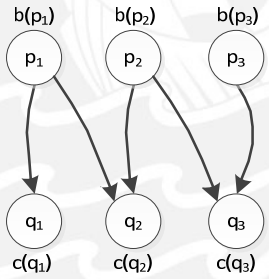

Fig. 1. 17: Red que represente un conjunto de proyectos y equipos

Este problema se puede modelar como uno de flujo máximo, para ello se agrega un nodo fuente que se conecte a cada uno de los proyectos con arcos de capacidad igual a la ganancia generada por el proyecto; y se agrega un nodo sumidero hacia el cual se conecten los equipos con arcos de capacidad igual al costo generado por usar dicho equipo; y por último, se coloca a los arcos que unen proyectos y equipos una capacidad igual a infinito. Se puede observar que el valor del flujo máximo de dicha red será el máximo valor de la ganancia que se puede obtener.

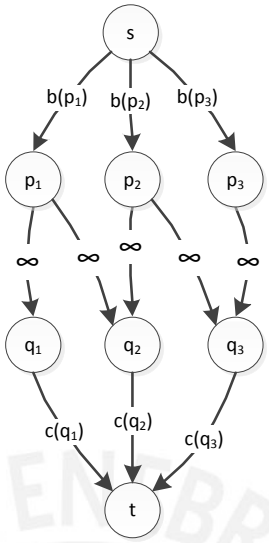

Fig. 1. 18: Problema de selección de proyectos como un problema de flujo máximo

1.3. Estado del arte

A continuación se detalla el estado del arte actual del problema del flujo máximo, es decir, los métodos de solución existentes de dicho problema y algunas aplicaciones de software que permiten calcular la solución al mismo.

1.3.1. Métodos de solución

El principal método de solución del problema del flujo máximo es el de Ford-Fulkerson, aunque también se han desarrollado algunas mejoras al mismo. También existen otros algoritmos que, aunque no calculan exactamente el valor del flujo máximo, calculan valores suficientemente cercanos al mismo, es decir, son métodos aproximados.

1.3.1.1. Método de Ford-Fulkerson

El método de Ford-Fulkerson, considerado un método más que un algoritmo (10), está basado en tres conceptos: red residual, trayectorias de aumento y cortes; así como en el teorema del flujo máximo/corte mínimo.

1.3.1.1.1. Red residual

La red residual es aquella que deriva de una red original y que está constituida por los arcos de dicha red que pueden admitir más flujo. A partir de este concepto se puede definir la capacidad residual de un arco, la cual es la cantidad de flujo restante que

todavía puede aceptar dicho arco. A los arcos que pertenecen a la red residual, se les denomina arcos residuales, además un arco (u,v) de una red puede aparecer en la red residual solo si por lo menos uno de los arcos (u,v) o (v,u) aparecen en la red original.

1.3.1.1.2. Trayectorias de aumento

Se denomina trayectoria de aumento a una trayectoria simple que va de la fuente al sumidero en la red residual. Por ende, cada uno de sus arcos admite cierta cantidad de flujo positivo sin violar la restricción de capacidad de los mismos. Además, la máxima cantidad en la que se puede incrementar el flujo en cada uno de los arcos de una trayectoria de aumento de tal manera que dicho incremento sea el mismo en todos los arcos, se denomina capacidad residual de la trayectoria de aumento.

1.3.1.1.3. Corte de red de flujo

Un corte es una división de la red de flujo en dos redes, una de las cuales contiene a la fuente y la otra al sumidero. La capacidad de un corte es igual a la suma de las capacidades de los arcos que van desde la parte que contiene a la fuente, hacia la parte que contiene al sumidero. Basado en estos conceptos, se denomina corte mínimo a aquel corte que posee la mínima capacidad entre todos los cortes que se le pueden hacer a la red de flujo. El corte mínimo es importante pues su capacidad limita la cantidad de flujo que puede circular a través de la red.

1.3.1.1.4. Teorema del flujo máximo-corte mínimo

Sea f un flujo que circula en una red de flujo $G = (V, E)$, con fuente *s* y sumidero *t*, f es el flujo máximo de G si:

1. La red residual de G no contiene trayectorias de aumento.
2. El valor de f es igual a la capacidad de alguno de los cortes de G.

Básicamente, el teorema establece que el flujo máximo que puede circular a través de una red es igual a la capacidad del corte mínimo de la misma. Este teorema es la base para el diseño del algoritmo de Ford-Fulkerson.

1.3.1.1.5. Algoritmo de Ford-Fulkerson

El algoritmo inicia otorgando un valor de flujo de cero a cada uno de los arcos de la red. Luego, se pasa a una fase iterativa, en la cual en cada repetición se busca una

trayectoria de aumento, y se incrementa el valor del flujo que circula en cada arco de dicha trayectoria por el valor de la capacidad residual de la misma (es decir, el flujo en esos arcos se incrementa en el menor valor de entre todas las capacidades de los arcos que constituyen la mencionada trayectoria), hasta que ya no se puedan encontrar trayectorias de aumento en la red. Entonces, por el teorema del flujo máximo – corte mínimo, una vez que ya no existan trayectorias de aumento el flujo máximo habrá sido calculado.

1.3.1.2. Método de Edmonds-Karp

El método de Edmonds-Karp es idéntico al de Ford-Fulkerson, con la diferencia de que la búsqueda de trayectorias de aumento está definida de manera que la trayectoria de aumento a encontrar sea la más corta entre la fuente y el sumidero, es decir, la trayectoria que tenga la menor cantidad de arcos (10). De esta manera, se logra una mejora en el tiempo de ejecución del algoritmo.

1.3.1.3. Método empujar-reetiquetar

En el método empujar-reetiquetar, en lugar de examinar toda la red residual para encontrar las trayectorias de aumento, se trabaja en un vértice a la vez y se realiza la búsqueda en vértices vecinos (10). Se basa en los conceptos de preflujo, excedente y altura del nodo.

1.3.1.3.1. Preflujo

El preflujo viene a ser el equivalente al concepto de flujo del método de Ford-Fulkerson, sin embargo diferencia de esta en la propiedad de conservación de flujo, pues en un preflujo esta propiedad se ve relajada. Así, dada una red $G = (V, E)$, con arcos con capacidad $c(u, v)$, un preflujo cumple las siguientes propiedades:

a. Restricción de capacidad: El flujo en un arco tiene valor numérico menor o igual a la capacidad del arco, es decir:

$$f(u, v) \leq c(u, v)$$

b. Simetría: El flujo en un arco puede representarse como un flujo de valor negativo en sentido inverso en el mismo arco, es decir:

$$f(u, v) = -f(v, u)$$

c. Relajación de la conservación de flujo: El flujo neto de un nodo (el flujo total que ingresa a un nodo menos el flujo total que sale del mismo) es un valor no negativo mayor o igual a cero (no necesariamente igual), es decir:

$$\sum f(X,v) - \sum f(v,Y) \geq 0$$

(12)

1.3.1.3.2. Excedente de un nodo

Se denomina excedente de un nodo ($e(v)$) al valor del flujo neto de dicho nodo; el cual, por la propiedad de relajación de conservación de flujo, es un valor no negativo no necesariamente igual a cero.

$$\sum f(X,v) - \sum f(v,Y) - e(v) \geq 0$$

(13)

1.3.1.3.3. Altura del nodo

La altura de un nodo ($h(v)$) es un valor numérico asignado a un nodo que permite determinar el sentido del flujo; así, el flujo circulará de nodos de mayor altura a nodos de menor altura, similar al desplazamiento del agua por los canales. Además, si $G = (V, E)$ es una red de flujo con fuente s y sumidero t, se establece:

$h(s) = |V|$

$h(t) = 0$

Donde $|V|$ es la cantidad de nodos de la red.

1.3.1.3.4. Algoritmo empujar-reetiquetar

El algoritmo empujar-reetiquetar se basa en dos operaciones fundamentales: el empuje y el reetiquetado.

1.3.1.3.4.1. Empuje

El empuje desde un nodo u a uno v significa enviar una parte o todo el excedente del nodo u hacia v. Para realizar un empuje se deben cumplir tres condiciones:

1. El excedente de u debe ser mayor que cero:

$e(u) > 0$

2. Debe existir capacidad disponible en el arco que va de u a v, es decir el arco (u,v) no debe estar completamente saturado o su capacidad residual debe ser mayor a cero:

$$c(u,v) - f(u,v) > 0$$

3. La altura del nodo u debe ser mayor al del nodo v:

$$h(u) > h(v)$$

Cumplidas estas condiciones, la cantidad a enviar desde u hacia v es el $min(e(u), c(u,v) - f(u,v))$.

1.3.1.3.4.2. Reetiquetado

Reetiquetar un nodo u significa incrementar su altura en por lo menos uno más que la de los nodos hacia los cuales dicho nodo puede enviar flujo, es decir, uno más que los nodos hacia los cuales va un arco que parte de u y que tiene capacidad disponible. Para realizar un reetiquetado se deben cumplir dos condiciones:

1. El excedente de u debe ser mayor que cero:

$$e(u) > 0$$

2. La altura de u es menor o igual a todas las alturas de los nodos hacia los cuales va un arco que parte de u y que tiene capacidad disponible:

$$h(u) \leq h(v); \forall \frac{v}{c(u,v)} - f(u,v) > 0$$

Cumplidas estas condiciones, se reetiqueta u al incrementar su altura en el menor valor que haga que $h(u) \leq h(v)$ para algún v donde $c(u,v) - f(u,v) > 0$.

1.3.1.3.4.3. Algoritmo empujar-reetiquetar

El algoritmo empujar-reetiquetar básicamente consiste primero en inicializar la altura y el excedente de cada vértice en cero, excepto en el sumidero, en el cual la altura será igual a la cantidad de nodos de la red; y el flujo de cada arco en cero. Luego se procede a realizar empujes y reetiquetados hasta que dichas operaciones ya no se puedan realizar. Una vez llegado a esto, el flujo máximo se habrá calculado.

1.3.2. Aplicaciones similares existentes

A continuación se describen algunas aplicaciones similares al software que se está desarrollando, ya existentes en el mercado global; para luego hacer un análisis comparativo entre las mismas.

1.3.2.1. Descripción de las aplicaciones

En esta parte se presenta una lista de algunas de las aplicaciones encontradas que resuelven el problema del flujo máximo.

1.3.2.1.1. Grafos

Grafos forma parte de un proyecto de investigación y desarrollo de aplicaciones informáticas orientadas hacia la docencia, investigación y labores profesionales en ingeniería y optimización (14).

1.3.2.1.1.1. Institución desarrolladora

El software Grafos fue desarrollado por un grupo de investigación en ingeniería y optimización a cargo del ingeniero Alejandro Rodríguez Villalobos, profesor de la Universidad Politécnica de Valencia (15).

1.3.2.1.1.2. Descripción de la aplicación

Grafos es un software diseñado para la construcción, edición y análisis de grafos; tiene como objetivo ser utilizado para la enseñanza y aprendizaje de la teoría de grafos en diversas áreas de estudio como la ingeniería, investigación de operaciones, diseño de redes y otras similares; así como para el modelado y resolución de problemas reales. (16)

1.3.2.1.1.3. Uso

La filosofía de grafos básicamente consiste en “dibujar, modelar, resolver y analizar”, por ello busca que el usuario tenga la absoluta libertad para tratar y abordar los problemas de grafos. Grafos permite dibujar libremente la red antes de preocuparse del problema que se va a resolver o el algoritmo que se necesitará para ello. Además, el

software reconoce alguna condición no factible o algún requerimiento faltante para la solución de algún problema, notificando al usuario de este hecho (16).

1.3.2.1.1.4. Esquema de trabajo

La versión actual del programa es la 1.2.9, y está estructurado de tal manera que cumpla con dos objetivos:

1. Poseer una interfaz que permita la fácil construcción y edición de redes en modo tabular o gráfico.
2. Poseer una estructura de clases y librerías que provean de diferentes algoritmos para la solución de determinados problemas.

1.3.2.1.1.5. Ventajas

- Posee una interfaz gráfica que facilita el ingreso de redes pequeñas y medianas.
- Posee varios algoritmos que permiten solucionar una amplia gama de problemas de investigación de operaciones.
- Libre, es decir, de distribución gratuita y de código abierto.

1.3.2.1.1.6. Desventajas

- No tiene manual de usuario.
- La interfaz gráfica no es muy eficiente con el ingreso de redes grandes, es decir, con un gran número de nodos y arcos (más de cien).

1.3.2.1.2. Graphing Calculator

Graphing Calculator es un software comercial que permite analizar problemas matemáticos gráficamente.

1.3.2.1.2.1. Institución desarrolladora

El software Graphic Calculator ha sido desarrollado por la empresa PacificTech (17).

1.3.2.1.2.2. Descripción de la aplicación

Graphic Calculator es un programa que permite visualizar objetos matemáticos en dos y tres dimensiones. Además permite crear gráficos animados, resolver ecuaciones gráficamente y escoger la perspectiva de observación de los objetos dibujados.

Adicionalmente, el software permite graficar funciones que estén dadas de manera explícita, implícita o parametrizada, tanto en dos o tres dimensiones.

1.3.2.1.2.3. Uso

Una de las características más resaltantes de Graphic Calculator es la simplicidad con la que se introducen los datos, pues no es necesario despejar las expresiones algebraicas con las que se va a trabajar, estas se introducen tal como están. Para el caso de problemas de investigación de operaciones, se debe introducir el problema en la forma general, es decir, maximizar o minimizar una función sujeta a un número determinado de restricciones (18).

1.3.2.1.2.4. Esquema de trabajo

La ventana principal del software está dividida en dos partes:

1. La primera permite el ingreso y edición de una o más expresiones algebraicas.
2. La segunda es la parte donde se grafican las mismas, a las cuales se le asigna un color para su respectiva identificación (18).

1.3.2.1.2.5. Ventajas

- Genera muy buenos gráficos que permiten visualizar claramente funciones en 2 y 3 dimensiones, y que permiten resolver gráficamente los problemas de investigación de operaciones.

1.3.2.1.2.6. Desventajas

- No utiliza el algoritmo de Ford-Fulkerson, resuelve los problemas de investigación de operaciones gráficamente.
- No es un software exclusivo para la investigación de operaciones, tiende más al cálculo y la geometría analítica.
- Tiene un costo considerable.

1.3.2.1.3. Mathematica

Mathematica es un software utilizado con fines científicos, de ingeniería, matemáticos y computacionales.

1.3.2.1.3.1. Institución desarrolladora

Mathematica fue originalmente concebida por el científico inglés Stephen Wolfram, quien continúa liderando el grupo investigador dedicado al desarrollo de dicho software en su compañía Wolfram Research (19).

1.3.2.1.3.2. Descripción de la aplicación

Mathematica es un poderoso software que permite visualizar gráficas en dos y tres dimensiones, manejar matrices, solucionar sistemas de ecuaciones; además brinda herramientas para el procesamiento de imágenes, estadística multivariable, minería de datos entre otras. Además ofrece un lenguaje de programación desarrollar soluciones a problemas complejos (20).

1.3.2.1.3.3. Uso

Básicamente, los problemas se ingresan de acuerdo a la sintaxis establecida por el software, para que este luego proceda a realizar los cálculos y emitir las respuestas a dichos problemas (19).

1.3.2.1.3.4. Esquema de trabajo

Mathematica se divide en dos partes fundamentales:

1. El kernel, el cual desempeña los cálculos a los problemas ingresados.
2. El frontend o interfaz, el cual despliega los resultados y permite al usuario interactuar con el kernel como si estuviera interactuando con un documento cualquiera.

1.3.2.1.3.5. Ventajas

- Gran cantidad de funcionalidades que facilitan el tratamiento y solución de problemas.

1.3.2.1.3.6. Desventajas

- Alto precio.
- No es un software exclusivo para la investigación de operaciones.

1.3.2.1.4. Invop

Este software pretende integrar los temas asociados a los modelos de redes, tanto en la teoría como en la práctica.

1.3.2.1.4.1. Institución desarrolladora

El software Invop fue desarrollado por la profesora Beatriz Loubet, docente de la Universidad Nacional de Cuyo en Argentina.

1.3.2.1.4.2. Descripción de la aplicación

El software tiene propósito netamente académico, es decir la enseñanza de la investigación de operaciones. Dentro de esta área el programa maneja los problemas de asignación, transporte, recorridos de redes y flujo. El software se encuentra en español y ofrece al usuario tanto la parte teórica como práctica para la solución de los problemas dados.

1.3.2.1.4.3. Uso

Básicamente, se elige el tipo de problema que se desea resolver, luego se ingresa los datos de la red a través de una matriz de datos; para luego solicitar el cálculo del resultado deseado.

1.3.2.1.4.4. Esquema de trabajo

El software Invop está dividido en cuatro funcionalidades principales, correspondientes a los cuatro tipos de problemas que resuelve:

1. Problema de asignación: Asignación óptima de recursos a tareas específicas.
2. Problema de transporte: Transporte de recursos desde centros de producción a centros de demanda.
3. Problema de distancia: Aquí están los problemas de ruta más corta, árbol de expansión mínima y agente viajero.
4. Problema de flujo: Flujo máximo (21).

1.3.2.1.4.5. Ventajas

- Gratuito
- Adecuado para la enseñanza

- Permite solucionar varios tipos de problemas de investigación de operaciones.

1.3.2.1.4.6. Desventajas

- Poco práctico para redes grandes.

1.3.2.1.5. WinQSB

WinQSB es un software interactivo de ayuda a la toma de decisiones que contiene herramientas muy útiles para resolver distintos tipos de problemas en el campo de la investigación operativa, con distintos módulos para tipo o modelo de problema dado (22).

1.3.2.1.5.1. Institución desarrolladora

WinQSB es un software de propiedad intelectual del Dr. Yih-Long Chang, experto en investigación de operaciones y profesor del Georgia Tech en Estados Unidos.

1.3.2.1.5.2. Descripción de la aplicación

WinQSBes un paquete de herramientas desarrollado para encontrar soluciones automatizadas a problemas complejos. El software incluye módulos para el análisis de muestreos, programación dinámica, elaboración de pronósticos, inventarios, procesos y cadenas de Markov, planificación de recursos, modelado de redes, programación no lineal, PERT, CPM, programación cuadrática, y otros problemas similares.

1.3.2.1.5.3. Uso

WinQSB utiliza los mecanismos típicos de las interfaces elaboradas en el sistema operativo Windows, es decir, con ventanas, menús desplegables, barras de herramientas, etc. Es por ello que el manejo del software es similar a cualquier otro que sea ejecutado en el entorno Windows.

1.3.2.1.5.4. Esquema de trabajo

El paquete incluye 19 módulos especializados para el planteo, análisis y solución de los problemas (23), entre los más importantes están:

- **Linear programming (LP) and integer linear programming (ILP):** Módulo encargado de resolver problemas de programación lineal gráficamente utilizando el

método de simplex, así como los de programación lineal entera con el procedimiento de ramificación y acotación.

- **Linear goal programming (GP) and integer linear goal programming (IGP):** Permite resolver problemas de programación multiobjetivo con restricciones lineales.
- **Quadratic programming (QP) and integer quadratic programming (IQP):** Permite resolver problemas de programación cuadrática con el método de simplex adaptado, y los de programación cuadrática entera con el procedimiento de ramificación y acotación.
- **Network modeling (NET):** Ayuda a resolver problemas de transbordo, de transporte, ruta más corta, flujo máximo, árbol generador y del agente viajero.
- **Nolinearprogramming (NLP)**: Permite resolver problemas de programación no lineal tanto los que no tienen restricciones (a través del método de búsqueda lineal) así como los que sí las presentan (con el método SUMT).
- **PERT – CPM:** Módulo que permite resolver problemas de secuencia de ejecución de actividades, los cuales son muy utilizados en la gestión de proyectos (22).

1.3.2.1.5.5. Ventajas

- Intuitivo, de entorno similar a cualquier otro que funcione en el sistema operativo Windows.
- Incluye módulos para la solución de una gran cantidad de problemas y modelos de investigación de operaciones.
- Entornos con funciones personalizables.
- Ejemplos y manuales adjuntos.

1.3.2.1.5.6. Desventajas

- No portable, solo funciona en Windows.
- Rudimentario.

1.3.2.1.6. Lingo

Lingo es una completa herramienta diseñada para la construcción y resolución de modelos de optimización lineal, no lineal y entera de manera fácil rápida y eficiente (24).

1.3.2.1.6.1. Institución desarrolladora

El software Lingo fue desarrollado por la empresa Lindo Systems Inc, la cual se ha caracterizado por proveer soluciones informáticas rápidas y fáciles de usar en el campo de la optimización matemática durante más de 21 años.

1.3.2.1.6.2. Descripción de la aplicación

LINGO (Linear, Interactive and General Optimizer) es un lenguaje de modelación matemática que provee un entorno en el cual se puede desarrollar, ejecutar y modificar modelos matemáticos. Es una herramienta fácil de utilizar para desarrollar grandes modelos de optimización lineal y no lineal (25).

El resultado que LINGO proporciona es una optimización que ayuda a encontrar la mejor solución: la de mayor ganancia o la de menor costo, problemas que generalmente involucran el uso eficiente de recursos (26).

1.3.2.1.6.3. Uso

Los grandes modelos de optimización usualmente se expresan como una función matemática a optimizar (maximizar o minimizar) sujeta a un número determinado de restricciones que mantienen una estructura similar. Dada esta característica, LINGO maneja los modelos como conjuntos (SET) de información y realizar las operaciones necesarias. Los SETs permiten definir y trabajar grupos de objetos que son procesados de manera similar, dichos objetos, a su vez, pueden también ser SETs. Con una definición de un conjunto se puede ingresar una serie de restricciones similares a través de sentencias simples y así formular expresiones cada vez más grandes y complejas (25).

1.3.2.1.6.4. Esquema de trabajo

Uno de los rasgos más útiles de LINGO es su aplicación en el lenguaje de modelado matemático, el cual permite expresar un problema de manera muy similar a la notación matemática normal y una serie de restricciones en una declaración compacta. Esto a su vez permite una mejor mantenibilidad de los modelos elaborados.

Otro aspecto importante es la sección de datos y variables, la cual está aislada de la formulación del modelo; es más, LINGO puede leer datos de hojas de cálculo

separadas, bases de datos o archivos de texto. Como los datos son independientes del modelo, se hace más sencillo realizar cambios y minimiza las posibilidades de incurrir en error cuando se elabora el modelo.

1.3.2.1.6.5. Ventajas

- Facilidad en la elaboración de los modelos.
- Permite cargar los datos de bases de datos u hojas de cálculo.
- Poderoso motor de soluciones.
- Amplia documentación y ayuda.

1.3.2.1.6.6. Desventajas

- No es libre.

1.3.2.2. Análisis comparativo entre aplicaciones

A continuación se presenta un análisis comparativo entre las soluciones previamente presentadas, en la que se incluye el software a desarrollar en el proyecto.

1.3.2.2.1. Tabla de análisis comparativo

Seguidamente se presenta la tabla del análisis comparativo entre las soluciones:

Nro.	Descripción	Peso	Grafos	Graphic Calculator	Mathematica	Invop	WinQSB	Lingo	Software a desarrollar
1	El software permite calcular el flujo máximo que puede circular en una red de flujo previamente dada con el uso del algoritmo de Ford-Fulkerson	5	X			X	X		X
2	El software permite el ingreso de la red de flujo y sus características vía archivos de texto	5						X	X

3	El software permite el ingreso de la red de flujo y sus características gráficamente	1	X	X	X				
4	El software permite visualizar los resultados del algoritmo en archivos de texto	5						X	X
5	El software manejará [illegible] específica para el ingreso de variables y ejecución de funciones y procedimientos.	5			X			X	X
6	El software permite visualizar los resultados del algoritmo gráficamente	1	X	X	X				
7	El software permite el procesamiento del problema como uno general de investigación de operaciones (con una función a optimizar y sus restricciones)	1		X				X	
8	El software permite solucionar los problemas con el método del simplex.	1		X	X			X	
9	El software puede ser utilizado con fines académicos de enseñanza de la investigación de operaciones y, en particular, del problema del flujo máximo.	5	X			X	X	X	X
10	El software permitirá resolver problemas de flujo máximo con redes de más de 100 nodos	5		X	X			X	X
11	El software permite resolver otros problemas de modelos de redes.	1	X	X	X	X	X	X	
12	El software permite resolver otros problemas de investigación de operaciones	1	X	X	X	X	X	X	
13	El software permite resolver problemas de otras áreas matemáticas además de la investigación de operaciones.	1		X	X			X	
14	El software permitirá el ingreso de los	2				X	X		

continuación

continúa en la página siguiente

	datos de la red por medio de matrices.								
15	El software puede ser ejecutado en cualquier plataforma o sistema operativo, es decir, es portable.	3							X
16	El software es libre y sin costo	5	X			X	X		X
	TOTAL	48	19	12	16	19	19	30	38

1.3.2.2.2. Conclusiones del análisis comparativo

Las conclusiones obtenidas del análisis comparativo se mencionan a continuación:

1. En el cuadro anterior se listó las principales características de los software estudiados y del desarrollado en el proyecto, a los cuales se les otorgó un peso según su importancia.
2. Se realizó un análisis comparativo entre las soluciones y se estableció las necesidades que satisface cada producto en función a las necesidades que se pretenden satisfacer con el software a desarrollar.
3. Se puede observar que cinco (Grafos, Graphic Calculator, Mathematica, Invop y WinQSB) de los seis productos estudiados obtuvieron un bajo puntaje en el análisis, esto se debe fundamentalmente a que dichos programas poseen una manera diferente a los archivos de texto para el ingreso de las variables de entrada y la salida de los resultados. Además, dos de ellos (Mathematica y Graphic Calculator), aparte de la investigación de operaciones, también han sido elaborados para resolver problemas de cálculo y otras ramas de la matemática.
4. Se ve también que el software Lingo es el de más alto puntaje, esto se debe a que este sí puede leer las entradas desde archivos de texto así como de hojas de cálculo y bases de datos, y emitir los resultados en modo texto; además maneja una sintaxis predefinida tanto para las entradas como para las salidas. Por lo tanto, este software es el más parecido al que se desea desarrollar, aunque con la desventaja de ser costoso y utilizar otro algoritmo para la solución del problema del flujo máximo.

1.4. Plan del proyecto

En esta parte se especifican las actividades y tareas a realizar en el presente proyecto.

1.4.1. Metodología de desarrollo

Para el presente proyecto se empleará el uso de una metodología basada en el modelo en cascada o ciclo de vida clásico, en el cual el trabajo fluye desde la identificación de requisitos hasta el despliegue del software de una manera casi lineal (2). Los pasos a seguir son la concepción, la elaboración (análisis y diseño) y la construcción (programación y pruebas). Además se hará uso de algunos artefactos de RUP que se consideren necesarios para el adecuado desarrollo del producto.

1.4.2. Necesidades de recursos

Para el desarrollo del presente proyecto se contará con los siguientes recursos:

1. Tesista.
2. Energía eléctrica.
3. Computadora con conexión a Internet.

1.4.3. Cronograma de actividades

ID	Task Name	Duration
1	**Proyecto de fin de carrera**	**150 days**
2	**Concepción**	**41 days**
3	**Elaboración del plan del proyecto**	**5 days**
4	Definición del problema	1 day
5	Definición de objetivos	1 day
6	Definición de resultados esperados	1 day
7	Definición de la metodología a utilizar	1 day
8	Planificación del proyecto	1 day
9	Elaboración del marco conceptual	5 days
10	**Elaboración del estado del arte**	**22 days**
11	Estudio de los métodos de solución del problema	10 days
12	Estudio de los productos existentes que brindan solución al problema	10 days
13	Análisis comparativo entre las soluciones	2 days
14	**Identificación de requerimientos**	**9 days**
15	Elaboración del catálogo de requisitos	3 days
16	Especificación de requisitos del software	6 days
17	**Elaboración**	**51 days**
18	**Análisis**	**3 days**
19	Elaboración del documento de visión del software	1 day
20	Elaboración del diagrama de clases de análisis	2 days
21	**Diseño**	**40 days**
22	Elaboración del estándar de diseño	1 day
23	Elaboración de la arquitectura del software	2 days
24	Elaboración de la arquitectura de la información	2 days
25	Elaboración de los diagramas de clases de diseño	4 days
26	Elaboración de los diagramas de secuencia	7 days
27	Elaboración del estándar de programación	1 day
28	Elaboración del estándar de interfaz gráfica	8 days
29	Elaboración del prototipo de pantallas	15 days
30	Elaboración del pseudocódigo del algoritmo a utilizar	2 days
31	**Diseño de pruebas**	**6 days**
32	Elaboración del plan de pruebas del software	1 day
33	Elaboración del catálogo de pruebas	5 days
34	**Construcción**	**38 days**
35	Implementación del algoritmo de Ford-Fulkerson	6 days
36	Implementación del software	30 days
37	Generación de instaladores	2 days
38	**Pruebas**	**20 days**
39	Ejecución de pruebas	10 days
40	Registro en la bitácora de pruebas	10 days

Fig. 1. 19. Cronograma de actividades del proyecto

1.4.4. WBS

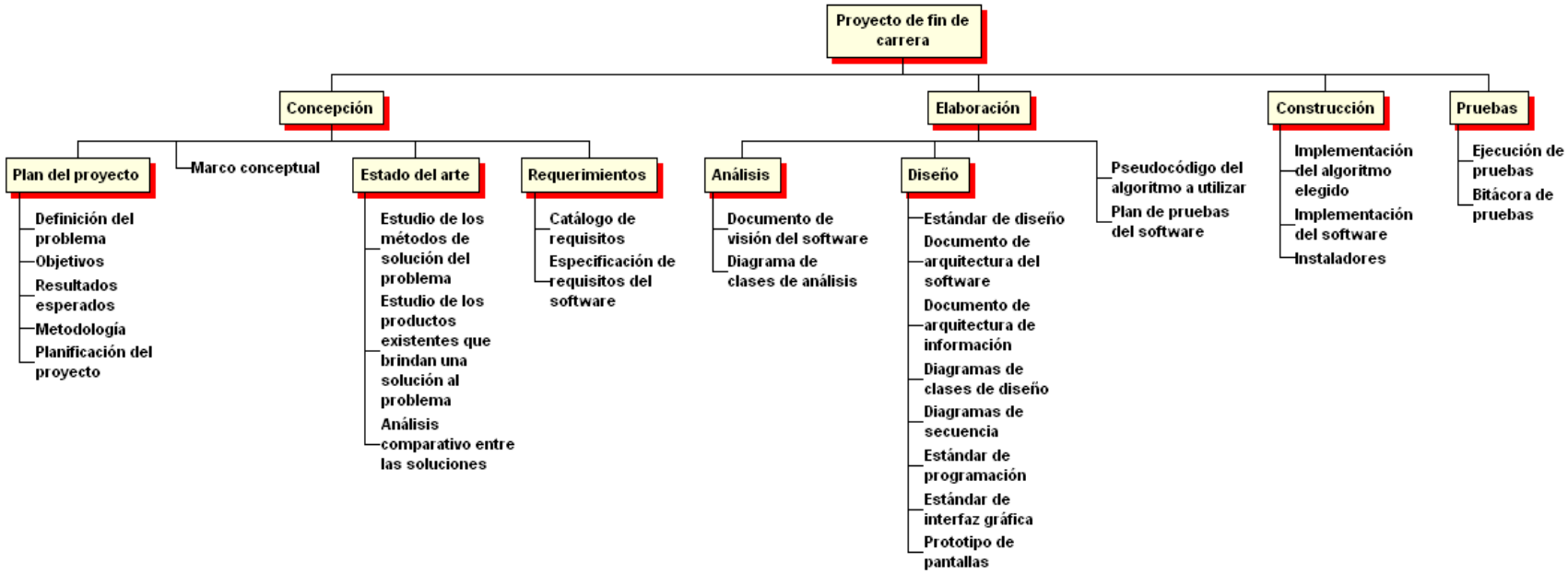

Fig. 1. 20. WBS del proyecto

1.5. Descripción y sustentación de la solución

A continuación se presentan el objetivo general del proyecto, los objetivos específicos, los resultados esperados, la sustentación de la solución y el alcance de la misma.

1.5.1. Objetivo general

Realizar el análisis, diseño e implementación de un software que permite obtener la solución automatizada al problema del flujo máximo, para lo cual se hará uso del algoritmo de Ford-Fulkerson.

1.5.2. Objetivos Específicos

1. Identificar los requisitos funcionales y no funcionales del software y realizar el análisis de los mismos.
2. Establecer los módulos que tendrá el software así como la interacción entre ellos.
3. Establecer las estructuras de datos a utilizar para la implementación del algoritmo.
4. Implementar el algoritmo de Ford-Fulkerson, y explicar los principales conceptos y teoremas en los cuales se basa.
5. Realizar la construcción y pruebas del software según el análisis y diseño previamente elaborados.

1.5.3. Resultados esperados

1. Documento de análisis del software.
2. Documento de arquitectura del software.
3. Descripción de las estructuras de datos a utilizar para la implementación del algoritmo
4. Estudio del algoritmo de Ford-Fulkerson y pseudocódigo del mismo.
5. Documentos de construcción y pruebas del software, así como el catálogo de pruebas a las cuales fue sometido el mismo.

1.5.4. Sustentación de la solución

Lo que se pretende en el presente proyecto es desarrollar un software libre pero de alta potencia que provea de una solución automatizada para situaciones de la vida real que pueden modelarse a través del problema del flujo máximo, y, adicionalmente, que

pueda ser utilizado con fines instructivos en la enseñanza de la investigación de operaciones.

Se busca que el software permita resolver el problema del flujo máximo en redes de arcos y nodos de manera automatizada, lo cual reducirá costos y tiempo en el contexto donde sea utilizado.

Al utilizar un software para encontrar solución a este problema, se superan las limitaciones en el tamaño de la red y tiempo de procesamiento, dicha red puede tener el tamaño adecuado para representar una situación real. Además, el tiempo en que el programa brinde una solución con el uso del algoritmo de Ford-Fulkerson será significativamente menor a que si se utilizaran métodos matemáticos convencionales de la investigación de operaciones, como el simplex.

1.5.5. Alcance

El software busca calcular de manera automática el flujo máximo en una red de arcos y nodos, cada arco con una capacidad máxima determinada.

Asimismo, los valores de las capacidades y flujos se está se están tomando como números enteros, pues los datos numéricos de dichas variables son fundamentalmente de ese tipo en la mayoría de sus aplicaciones.

El objetivo del proyecto se centra principalmente en el estudio del problema del flujo máximo, así como el análisis e implementación del algoritmo de Ford-Fulkerson para la obtención de la solución de dicho problema, por ello las aplicaciones particulares del problema en contextos específicos serán tocados con menor profundidad, y se dejarán como investigaciones en trabajos futuros. Asimismo, otros temas importantes de investigación de operaciones como son el problema de transporte, la programación de tareas, la ruta crítica y otros también serán mencionados para implementarse como trabajos futuros.

Además, al manejar redes de gran tamaño (más de 100 nodos), resulta poco práctico utilizar un constructor gráfico para el ingreso de las mismas, por lo cual dicha parte se dejará para futuras investigaciones.

Capítulo 2: Análisis

A continuación se realiza el análisis del software, en el cual se establece primero la metodología a seguir, se establecen los requerimientos y se efectúa el análisis propiamente dicho.

2.1. Metodología

Para la ejecución del presente proyecto se utiliza una metodología derivada del modelo en cascada o ciclo de vida clásico, según el cual, el trabajo se realiza linealmente desde la identificación de requisitos hasta el despliegue del software (2). Se toma esta opción pues la cantidad de recursos empleados para el proyecto (un solo desarrollador: el tesista) facilita en gran manera la planeación, coordinación e integración del software así como de la documentación que lo sustenta. Además se utilizan algunos de los principales artefactos recomendados por RUP, necesarios para el ordenado y adecuado desarrollo del producto, principalmente porque el tamaño del software hace innecesario el uso de toda la gama de procedimientos señalados por dicha metodología.

Las etapas y actividades desarrolladas para el proyecto son detalladas a continuación:

2.1.1. Concepción

En esta etapa se define el objetivo del proyecto y se determinan los principales procesos del mismo. Además, se definen los requerimientos que cumple el software, tanto funcionales como no funcionales. Esta etapa consta de las siguientes actividades:

a. Elaboración del plan de proyecto (ver Capítulo 1)

- Definición del problema.
- Definición de objetivos.
- Definición de resultados esperados.
- Definición de la metodología a utilizar.
- Planificación del proyecto.

b. Elaboración del marco conceptual (ver Capítulo 1)

c. Elaboración del estado del arte (ver Capítulo 1)

- Estudio de los métodos de solución del problema.
- Estudio de los productos existentes que brindan una solución al problema.
- Análisis comparativo entre las soluciones.

d. Identificación de requerimientos:

- Elaboración del catálogo de requisitos del software (ver punto 2.2).
- Especificación de requisitos del software (ver Anexo B).

2.1.2. Elaboración

Aquí se realiza el análisis y diseño del software y de las pruebas necesarias para la verificación y validación del mismo. Para esta etapa se realizan las siguientes actividades:

a. Análisis:

- Elaboración del documento de visión del software (ver Anexo A).
- Elaboración del diagrama de clases de análisis (ver punto 2.3.3. y 2.3.4.).

b. Diseño

- Elaboración del estándar de diseño (Anexo E).
- Elaboración del documento de arquitectura del software (ver punto 3.1.).
- Elaboración del documento de arquitectura de información (ver punto 3.3.).
- Elaboración del diagrama de clases de diseño (ver Anexo E).

- Elaboración de diagramas de secuencia (ver Anexo E).
- Elaboración del estándar de programación (ver Anexo D).
- Elaboración del estándar de interfaz gráfica (ver punto 3.2.).

c. Pseudocódigo del algoritmo (ver punto 4.2.)

d. Diseño de pruebas

- Elaboración del plan de pruebas del software (ver punto 4.3.).
- Elaboración del catálogo de pruebas (ver Anexo G).

2.1.3. Construcción

En esta etapa se realiza la implementación del software según todos los documentos antes mencionados. Esta etapa consta de las siguientes actividades:

a. Implementación del algoritmo

b. Implementación del software

c. Generación de instaladores

2.1.4. Pruebas

Finalmente, aquí se ejecutarán las pruebas según el plan de pruebas del software y el catálogo de pruebas del mismo. Para esta etapa se realizan las siguientes actividades:

a. Ejecución de pruebas.

b. Registro en la bitácora de pruebas (ver Anexo G).

2.2. Requerimientos

A continuación se presentan los requisitos que debe cumplir el software a implementar en el presente proyecto.

2.2.1. Requerimientos funcionales

No.	Descripción	Prioridad
1	El software a desarrollar permitirá el ingreso de la red en la cual se desea resolver el problema del flujo máximo.	1
2	El software a desarrollar permitirá almacenar la red ingresada y sus características para futuras visualizaciones.	1

continúa en la siguiente página

continuación

3	El software a desarrollar permitirá cargar redes existentes previamente guardadas y poder modificarlas o calcular el flujo máximo que circula por él.	1
4	El software a desarrollar permitirá comprobar la validez del archivo que contiene las características de la red (ya sea de texto o xml), así como de sus nodos y arcos, a fin de verificar el cumplimiento de las condiciones necesarias para la aplicación del método de solución elegido.	1
5	El software permitirá ingresar redes con más de una fuente o más de un sumidero y resolverlos sin complicación adicional.	1
6	El software a desarrollar permitirá calcular el flujo máximo que puede circular por la red ingresada así como el flujo sobre cada uno de los arcos de la misma con el uso del algoritmo de Ford – Fulkerson.	1
7	El software a desarrollar permitirá mostrar y almacenar los resultados de la ejecución del algoritmo en archivos de texto y xml.	1

2.2.2. Requerimientos no funcionales

No	Descripción	Tipo	Prioridad
1	Se utilizará lenguaje Java, con el IDE Netbeans 6.7.1	Implementación	1
2	Los resultados de la ejecución del algoritmo de cálculo de caudales podrán almacenarse en archivos de texto y XML.	Implementación	1
3	La aplicación podrá funcionar sobre sistemas operativos Windows, Linux o MacOS, con lo que se aprovechará la portabilidad ofrecida por Java.	Implementación	1

2.2.3. Prioridades

Número	Descripción
1	Alta
2	Media
3	Baja

2.2.4. Restricciones

- La función principal de la aplicación es obtener el flujo máximo que puede circular por una red o grafo, así como el flujo que va por cada uno de los arcos del mismo.
- La aplicación puede trabajar con redes muy grandes, de gran cantidad de nodos y arcos, por lo que la manera ideal de ingresar las mismas es a través de archivos planos, maneras alternativas como gráficos o matrices se dejan para futuras investigaciones.

2.2.5. Reglas del negocio

- El problema del flujo máximo es uno de los más importantes modelos de investigación de operaciones diseñados y su uso se extiende a un gran número de aplicaciones en la vida real.
- En este contexto, el flujo es un concepto general que toma diferentes acepciones de acuerdo al problema específico que se trata a desarrollar con este modelo (por ejemplo, el flujo puede referirse a volumen de líquidos, cantidad de objetos circulantes, etc). Más información puede obtenerse en el Capítulo 1, punto 1.2.4. Aplicaciones del problema de flujo máximo.

2.3. Análisis

A continuación se presenta el análisis realizado sobre los requerimientos establecidos para el software a desarrollar.

2.3.1. Modelo de casos de uso

El modelo de casos de uso establece las interacciones entre el usuario y el software, así como las funcionalidades que este último presenta.

2.3.1.1. Catálogo de actores

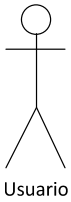

Fig. 2. 1. Diagrama de actores

Usuario

Representa a cualquier usuario del software

2.3.1.2. Módulos del software

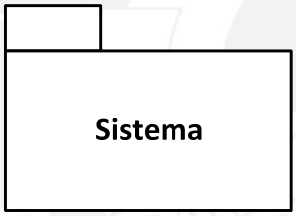

Fig. 2. 2. Diagrama de módulos del software

Dado el tamaño del software a implementar, resulta suficiente englobar el mismo en un solo módulo que incluya toda la interacción con el sistema, desde el ingreso de los datos, el procesamiento de los mismos, y la emisión y visualización de resultados.

2.3.1.3. Casos de uso

Seguidamente se listan los casos de uso identificados para con el software

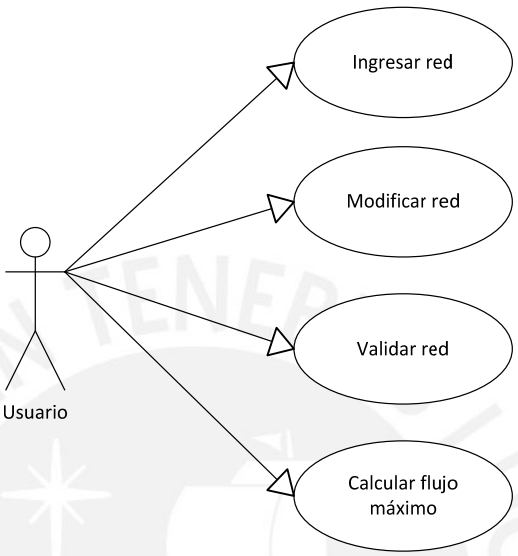

Fig. 2. 3. Diagrama de casos de uso del software

Ingresar red

Este caso de uso se refiere a la creación de la red por el usuario vía archivos de texto o xml.

Modificar red

Este caso de uso referencia a la posibilidad que tiene el usuario de abrir una red previamente creada para su evaluación o modificación.

Validar red

Este caso de uso se refiere a la validación de la sintaxis del archivo de texto o xml que contiene la descripción de una red que se desea evaluar.

Calcular flujo máximo

Este caso de uso se refiere a la ejecución del algoritmo de Ford-Fulkerson por parte del software para el cálculo del flujo máximo y los flujos individuales que circulan por cada uno de los arcos de la red.

2.3.2. Características de los usuarios

Los usuarios del software son personas con conocimiento teórico – práctico sobre modelos de redes y el problema del flujo máximo, capaces de interpretar los resultados obtenidos y utilizarlos de manera adecuada.

2.3.3. Diagrama de clases de análisis

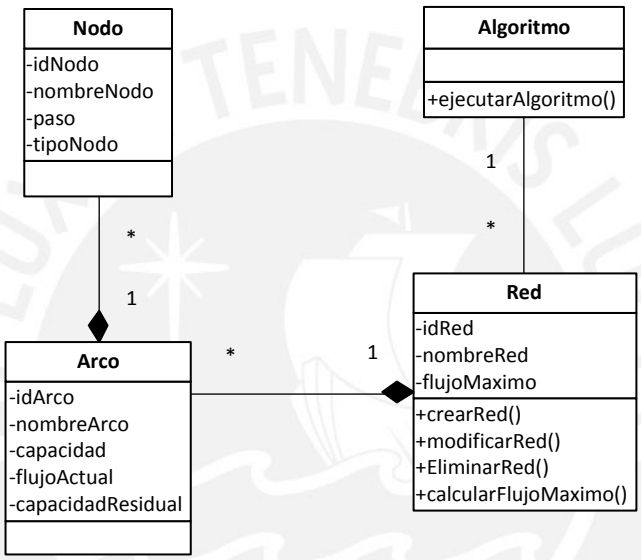

Fig. 2. 4. Diagrama de clases de análisis

2.3.4. Diccionario de clases de análisis

Algoritmo	
Clase que contiene al algoritmo que obtendrá la solución al problema planteado	
Operación	**Descripción**
ejecutarAlgoritmo	Operación que permite la ejecución del algoritmo de Ford-Fulkerson para la solución del problema del flujo máximo.

Red
Representa a la red de arcos y nodos a analizar

Atributo	Descripción
idRed	Código identificador de la red
nombreRed	Nombre de la red
flujoMaximo	Flujo máximo capaz de circular por la red
Operación	**Descripción**
registrarRed	Ingreso de una nueva red (de sus nodos y arcos)
modificarRed	Modificación de una red existente (agregar nuevos nodos o arcos o modificar los datos de los existentes)
eliminarRed	Eliminación de una red existente
calcularFlujoMaximo	Operación que permite calcular el flujo máximo capaz de circular en la red, así como los flujos individuales que van por cada uno de los arcos

Arco	
Representa un arco de la red.	
Atributo	**Descripción**
idArco	Código identificador del arco
nombreArco	Nombre del arco
capacidad	Capacidad máxima de flujo que el arco puede soportar
flujoActual	Flujo que actualmente circula por dicho arco
capacidadResidual	Cantidad de flujo restante que un arco puede recibir cuando ya circula por él un flujo determinado. Matemáticamente es la diferencia entre la capacidad y el flujo actual del arco.

Nodo
Representa un nodo de la red

Atributo	Descripción
idNodo	Código identificador del nodo en la red
nombreNodo	Nombre del nodo
paso	Indicador que advierte si el flujo pasa o no por el nodo.
tipoNodo	Tipo de nodo: fuente, sumidero o simple.

Capítulo 3: Diseño

A continuación se realiza el diseño del software, en el cual se establece la arquitectura de la solución, el diseño de la interfaz gráfica y la arquitectura de la información.

3.1. Arquitectura de la solución

En este punto se describen las principales características de la arquitectura empleada para el diseño del software

3.1.1. Representación de la arquitectura

Los estilos arquitectónicos que sigue el software son:

3.1.1.1. Orientado a Objetos.

Las entidades de negocio, la lógica del mismo y los módulos serán implementados teniendo como base el modelo orientado a objetos.

3.1.1.2. Orientado a Eventos.

Los controles de Java y todas las acciones ejecutadas por el software utilizarán eventos para su interacción con el usuario.

3.1.1.3. Repositorio.

Para el desarrollo del sistema se emplearán archivos de texto y xml, los cual servirán de repositorios para los datos de entrada y la información de salida.

3.1.1.4. Monolítica

El software está integrado en una sola aplicación que incluirá el acceso a las funcionalidades brindadas por el mismo, por ende, la instalación y funcionamiento del software se ejecutará en la máquina del cliente. Asimismo, si se desea utilizar el software en varias terminales, bastará con instalarlo en todos los terminales donde se trabaje.

3.1.1.5. La arquitectura se muestra como una serie de vistas.

Las vistas que se tienen son: Vista de casos de uso, Vista de procesos, Vista de despliegue, Vista lógica y Vista de implementación. Estas vistas son presentadas usando Microsoft Visio y empleando UML.

3.1.2. Metas y restricciones de la arquitectura

El software está implementado bajo una arquitectura monolítica multicapas (utilizará cinco capas). La primera capa estará orientada a representar las entidades del negocio, la segunda a la interfaz de usuario, la tercera a la aplicación que se encarga de gestionar la lógica del negocio y la última al almacenamiento de datos en objetos persistentes (archivos).

3.1.3. Vista de casos de uso

Los casos por módulo de uso pueden visualizarse en el punto 2.3.1.3, figura 2.3.

3.1.4. Vista de procesos

La vista de procesos muestra las tareas o procesos que se efectúan con el sistema y la forma en la que se comunican los mismos. Para dicha finalidad se hace uso del diagrama de actividades.

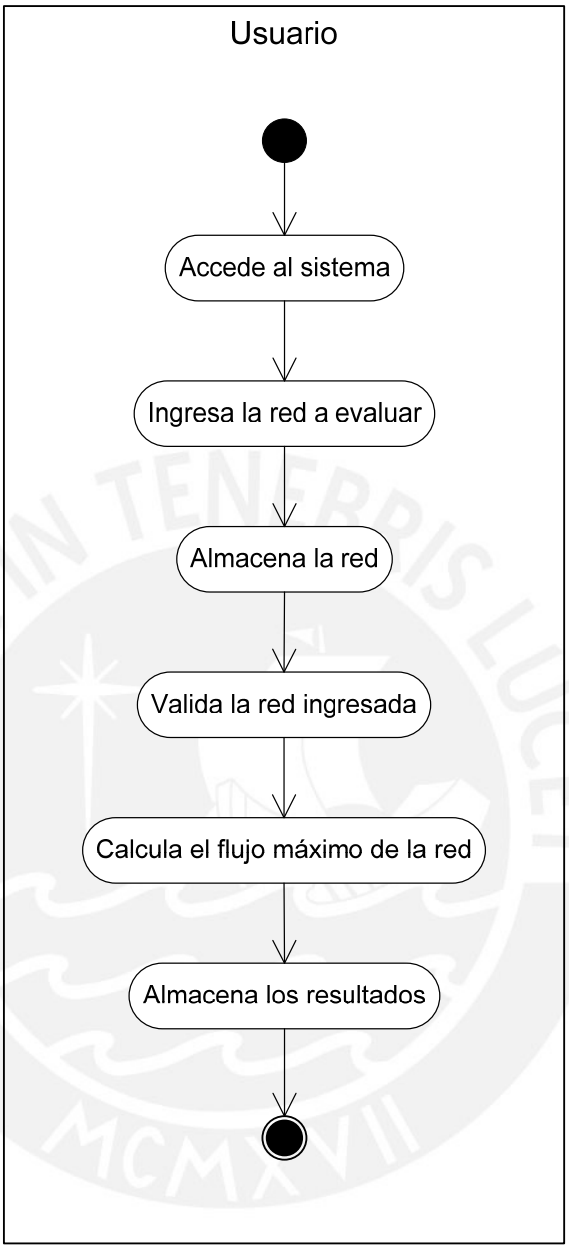

Fig. 3. 1. Diagrama de actividades

3.1.5. Vista lógica

La vista lógica de la arquitectura describe las clases más importantes, su organización en paquetes de servicio y subsistemas, y la organización de estos en capas.

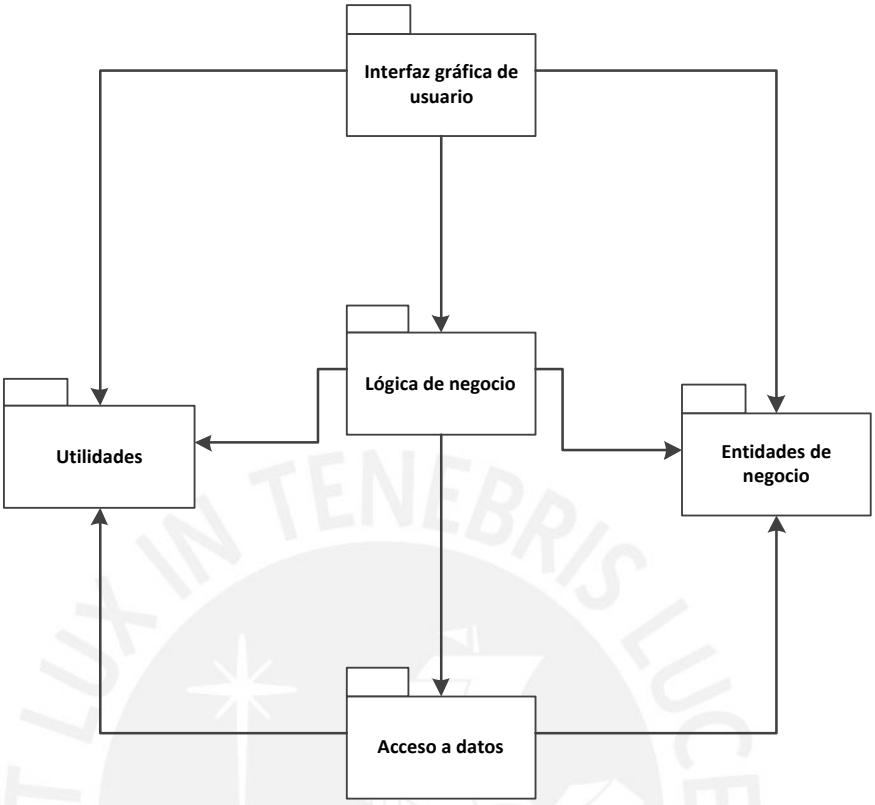

Fig. 3. 2. Diagrama de paquetes y la interrelación entre ellos

3.1.5.1. Interfaz Gráfica de Usuario.

La capa de Interfaz Gráfica de Usuario contiene todas las clases que representan los formularios que el usuario puede ver y con los cuales puede interactuar. Esta capa depende de la Lógica del Negocio, además separa al usuario de la capa media del sistema.

3.1.5.2. Lógica del Negocio.

Los servicios del negocio están representados por las clases controladoras que se encargan de manejar las entidades de negocio de la aplicación, por lo que depende de la capa de Entidades de Negocio.

3.1.5.3. Entidades del Negocio.

La capa de Entidades del Negocio contiene a todas las clases que representan las entidades en el dominio de la aplicación.

3.1.5.4. Acceso a Datos.

La capa de Acceso de Datos contiene a todas las clases que nos permitirán tener acceso a los archivos persistentes e interactuar con ellos.

3.1.5.5. Utilidades.

Esta capa contiene funcionalidades adicionales que asistirán a las clases de la Interfaz Gráfica de Usuario, de la Lógica del Negocio e incluso del Acceso a Datos.

3.1.6. Vista de despliegue

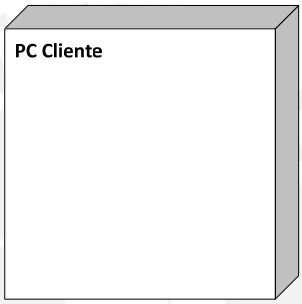

Fig. 3. 3. Vista de despliegue del software

3.1.6.1. PC Cliente.

Mediante este tipo de computador los usuarios acceden al sistema y pueden hacer uso de parte o toda la funcionalidad que el software provee.

3.1.7. Vista de implementación

Esta sección describe la estructura general de la implementación del modelo.

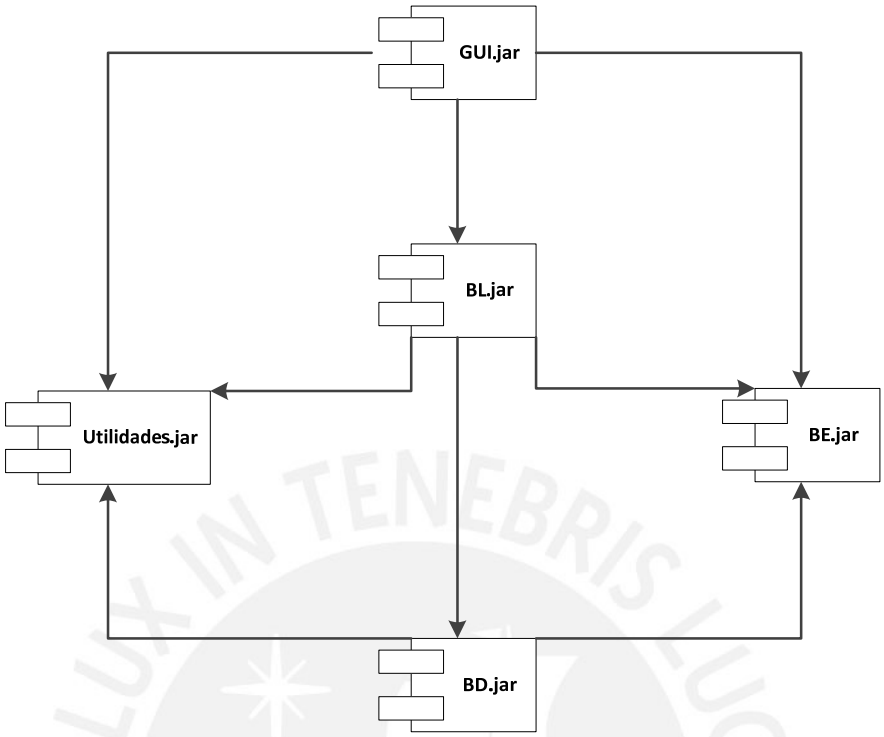

Fig. 3. 4. Vista de implementación del software

3.1.7.1. Interfaz gráfica de usuario.

Componente representado por la librería GUI.jar que se encarga de manejar las interfaces gráficas con las que el usuario interactúa. Implementa la capa Interfaz Gráfica de Usuario.

3.1.7.2. Lógica del negocio.

Componente representado por la librería BL.jar, que se encarga de las tareas relacionadas con los procesos de diseño y tratamiento del problema del flujo máximo. Implementa la capa Lógica del Negocio.

3.1.7.3. Entidades del negocio.

Componente representado por la librería BE.jar, que se encarga del manejo de las entidades relacionadas con el problema del flujo máximo. Implementa la capa Entidades del Negocio.

3.1.7.4. Objetos de conexión a Base de Datos.

Componente representado por la librería BD.jar, que se encarga del acceso a los archivos que constituyen la data persistente. Implementa la capa Acceso a Datos.

3.1.7.5. Utilidades.

Componente representado por la librería Utilidades.jar, que se encarga del manejo de utilidades y servicios adicionales para el software. Implementa la capa Utilidades.

3.1.8. Vista complementaria de resumen

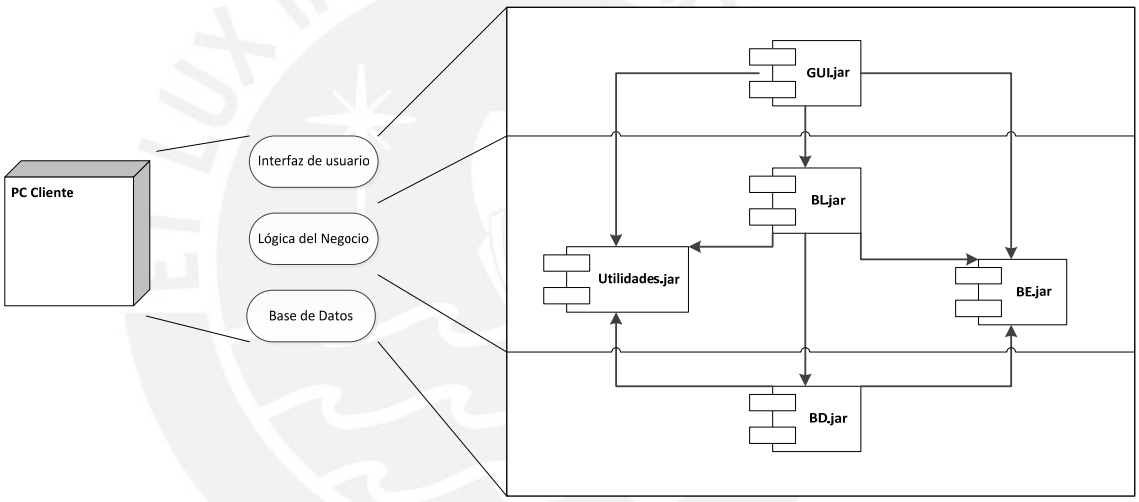

Fig. 3. 5. Vista complementaria resumen de la arquitectura

3.1.9. Calidad

Aquí se definen los rangos de la calidad para el funcionamiento, robustez, tolerancia a fallos, utilidad y demás características similares del software a desarrollar

3.1.9.1. Disponibilidad:

El software estará disponible cada vez que un usuario lo requiera.

3.1.9.2. Capacidad de mantenimiento:

El software será diseñado para permitir facilidad de mantenimiento, respetando el diseño de la interfaz lo más que se pueda.

3.2. Diseño de la interfaz gráfica

A continuación se describen los principales criterios que rigen la interfaz gráfica de usuario del software. En consecuencia, las pantallas obedecen lo establecido en este punto. Dichas pantallas pueden observarse en el Anexo F.

3.2.1. Pantalla principal

La pantalla principal está constituida por un editor de texto donde se puede ingresar los archivos de entrada y donde se visualiza el resultado o salida del programa. Este editor tendrá tres partes bien diferenciadas:

1. Barra de menú: Contiene las opciones básicas para el manejo de los archivos, su contenido y el procesamiento de las redes para el cálculo del flujo máximo
2. Editor de texto: Área asignada para el ingreso de los archivos de entrada y visualización de los archivos de salida
3. Caja de texto de mensajes de error: Área no editable donde se muestran los mensajes de error en la validación y procesamiento de los archivos de entrada.

A continuación se muestra el prototipo de la pantalla principal donde se indican las tres partes antes mencionadas.

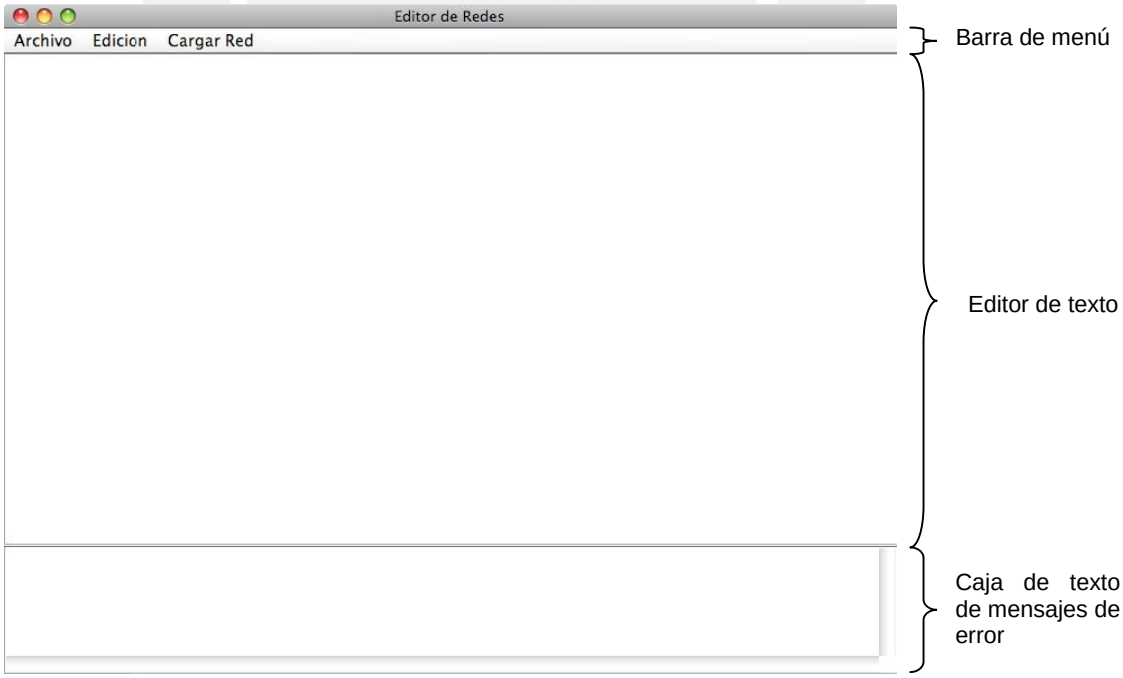

Fig. 3. 6. Pantalla principal del software

3.2.1.1. Barra de menú

Fig. 3. 7. Barra de menú del software

La barra de menú contiene tres opciones:

Menú	Descripción
Archivo	Contiene las opciones para la administración de los archivos de entrada y salida utilizados para el cálculo del flujo máximo
Edición	Contiene las opciones para la edición de un archivo en particular.
Cargar Red	Contiene las opciones para la validación de los archivos de entrada y el cálculo del flujo máximo.

1. Menú Archivo

Fig. 3. 8. Menú Archivo

Opción	Descripción
Nuevo	Abre un nuevo archivo en blanco para el ingreso de un nuevo archivo de entrada de la red
Abrir	Abre un archivo que contiene una red existente
Guardar	Almacena la información del archivo que actualmente se está trabajando
Guardar como …	Almacena la información trabajada en un nuevo archivo
Imprimir	Imprime el archivo de la red
Salir	Cierra el programa

2. Menú Edición

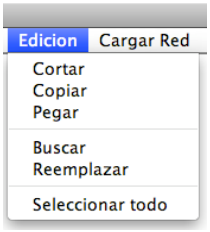

Fig. 3. 9. Menú Edición

Opción	Descripción
Cortar	Corta el texto seleccionado del editor de texto
Copiar	Copia el texto seleccionado del editor de texto
Pegar	Pega el texto copiado o cortado en la posición actual en el que se encuentra el cursor en el editor de texto.
Buscar	Busca un texto en el editor de texto.
Reemplazar	Reemplaza un texto por otro en el editor de texto.
Seleccionar todo	Selecciona todo el texto del editor de texto.

3. Menú Cargar Red

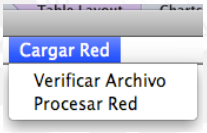

Fig. 3. 10. Menú Cargar Red

Opción	Descripción
Verificar Archivo	Opción que permite validar si el archivo de entrada tiene el formato correcto.
Procesar Red	Opción que permite calcular el flujo máximo que pasa por la red ingresada y los flujos individuales que pasan por cada arco respectivo.

3.2.1.2. Editor de texto

El editor de texto es el área destinada al ingreso de los archivos de entrada y visualización de los archivos de salida con los resultados del algoritmo. A continuación se presenta un ejemplo del editor con un archivo de entrada.

Fig. 3. 11. Editor de texto con archivo de entrada

3.2.1.3. Caja de texto de mensajes de error

En esta área se mostrarán los mensajes de error de formato cuando se valide los archivos de entrada del software.

Fig. 3. 12. Caja de texto de mensajes de error con mensaje

3.3. Arquitectura de la información

En este punto se detalla la forma cómo fluye la información en el software, así como la estructura de los archivos que se utilizan para almacenar tanto las redes ingresadas, como las resultantes.

3.3.1. Flujo de información del software

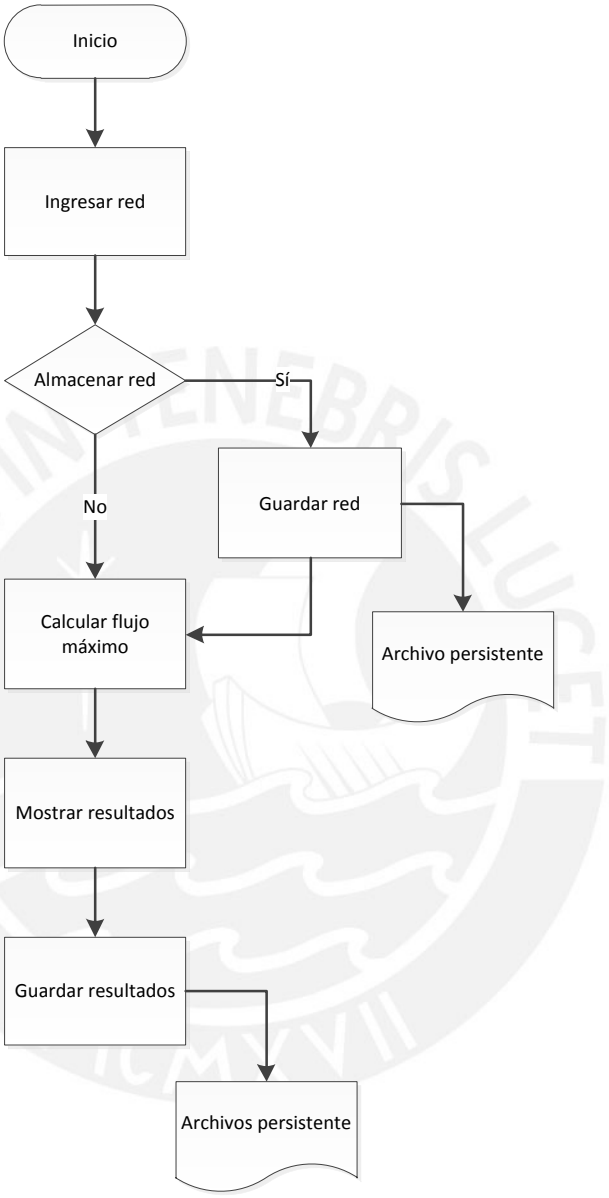

Fig. 3. 13. Flujo de información del Software

3.3.2. Persistencia

La información de las redes sobre las cuales se realizará el cálculo del flujo máximo y los flujos circulantes por cada arco se proveerá a través de archivos xml y de texto; asimismo, la información resultante que emita el software también se almacena en archivos de este tipo.

3.3.2.1. Archivos de texto

Los archivos de texto de entrada (input) obedecerán el siguiente formato:

- Para listar los nodos:

  Los nodos se diferencian según sean nodos fuentes, intermedios o sumideros según los siguientes títulos:

  1. NodosFuentes
  2. NodosIntermedios
  3. NodosSumideros

  Asimismo existirá un espacio en blanco al final de cada tipo de nodo.

  Cada nodo se listará con el siguiente formato: IdNodo=<identificador del nodo>

  Cada nodo se colocará en una sola línea. Asimismo en ningún caso el identificador del nodo puede ser cero.

- Para listar los arcos

  Etiqueta: Arcos

  Cada arco se lista con el siguiente formato: Arco=<identificador del nodo fuente>;<identificador del nodo destino>;capacidad=<capacidad del arco>

  Cada arco se coloca en una sola línea

A continuación se presenta un ejemplo del archivo de entrada de seis nodos y diez arcos:

```
NodosFuentes
idNodo=1
idNodo=2

NodosIntermedios
idNodo=3
idNodo=4

NodosSumideros
idNodo=5
idNodo=6
```

Arcos
Arco=1;2;capacidad=16
Arco=1;3;capacidad=13
Arco=2;3;capacidad=10
Arco=2;4;capacidad=12
Arco=3;2;capacidad=4
Arco=3;5;capacidad=14
Arco=4;3;capacidad=9
Arco=4;6;capacidad=20
Arco=5;4;capacidad=7
Arco=4;5;capacidad=4
Arco=4;6;capacidad=12

Similarmente, los archivos de texto de salida (output) obedecerán el siguiente formato:

- Valor del flujo máximo indicado por la etiqueta “Flujo máximo=<valor del flujo>”
- Para listar los nodos:
  Los nodos se diferencian según sean nodos fuentes, intermedios o sumideros según los siguientes títulos:
  1. NodosFuentes
  2. NodosIntermedios
  3. NodosSumideros

  Asimismo existirá un espacio en blanco al final de cada tipo de nodo.
  Cada nodo se listará con el siguiente formato: IdNodo=<identificador del nodo>
  Cada nodo se colocará en una sola línea.

- Para listar los arcos
  Etiqueta: Arcos
  Cada arco se lista con el siguiente formato: Arco=<identificador del nodo fuente>;<identificador del nodo destino>;capacidad=<capacidad del arco>;flujo=<valor del flujo que circula por dicho arco>
  Cada arco se coloca en una sola línea

A continuación se presenta un ejemplo del archivo de salida de seis nodos y diez arcos:

```
Flujo máximo=26

NodosFuentes
idNodo=1
idNodo=2

NodosIntermedios
idNodo=3
idNodo=4

NodosSumideros
idNodo=5
idNodo=6

Arcos
Arco=1;2;capacidad=16;flujo=16
Arco=1;3;capacidad=13;flujo=10
Arco=2;3;capacidad=10;flujo=8
Arco=2;4;capacidad=12;flujo=12
Arco=3;2;capacidad=4;flujo=4
Arco=3;5;capacidad=14;flujo=14
Arco=4;3;capacidad=9;flujo=0
Arco=4;5;capacidad=4;flujo=4
Arco=4;6;capacidad=12;flujo=8
Arco=5;4;capacidad=7;flujo=0
```

3.3.2.2. Archivos XML

Para los archivos XML, estos se implementarán en base a las especificaciones establecidas por el lenguaje GraphXML, el cual permite expresar la estructura de una red en lenguaje XML (27), sin embargo, dicha especificación será adaptada a los objetivos y restricciones del proyecto.

Los archivos XML de entrada obedecerán la estructura del siguiente DTD:

```
<!DOCTYPE input [
        <!ELEMENT input(NodosFuentes | NodosIntermedios | NodosSumideros |
        Arcos)*>
                <!ELEMENT NodosFuentes (Nodo)>
                        <!ELEMENT Nodo EMPTY>
                                <!ATTLIST Nodo idNodo CDATA #REQUIRED>
                <!ELEMENT NodosIntermedios (Nodo)>
                        <!ELEMENT Nodo EMPTY>
                                <!ATTLIST Nodo idNodo CDATA #REQUIRED>
                <!ELEMENT NodosSumideros (Nodo)>
                        <!ELEMENT Nodo EMPTY>
                                <!ATTLIST Nodo idNodo CDATA #REQUIRED>
                <!ELEMENT Arcos (Arco)*>
                        <!ELEMENT Arco EMPTY>
                                <!ATTLIST Arco idNodoInicio CDATA #REQUIRED>
                                <!ATTLIST Arco idNodoFin CDATA #REQUIRED>
                                <!ATTLIST Arco capacidad CDATA #REQUIRED>
]>
```

Donde:

Elemento “input”: El elemento input es el elemento base que contiene el resto de elementos del archivo XML.
Elemento “NodosFuentes”: Representa el conjunto de nodos fuente de la red.
Elemento “Nodo”: Representa un nodo fuente.
Atributo “idNodo” del elemento “Nodo”: Valor del código del nodo fuente.
Elemento “NodosIntermedios”: Representa el conjunto de nodos intermedios, es decir, los nodos que no son ni fuente ni sumidero, de la red.
Elemento “Nodo”: Representa un nodo intermedio.
Atributo “idNodo” del elemento “Nodo”: Valor del código del nodo intermedio
Elemento “NodosSumideros”: Representa el conjunto de nodos sumideros de la red.

Elemento “Nodo”: Representa un nodo sumidero.
Atributo “idNodo” del elemento “Nodo”: Valor del código del nodo sumidero.
Elemento “Arcos”: Representa el conjunto de arcos de la red
Elemento “Arco”: Representa un arco de la red.
Atributo “idNodoInicio” del elemento “Arco”: Valor del código del nodo de donde parte la red.
Atributo “idNodoFin” del elemento “Arco”: Valor del código del nodo destino
Atributo “capacidad” del elemento “Arco”: Valor de la capacidad del arco.

Cabe destacar que en ningún caso el identificador del nodo puede ser cero.

Así tenemos por ejemplo:

```
<?xml version=”1.0”?>
<input>
<NodosFuentes>
<Nodo idNodo="1" />
</NodosFuentes>
<NodosIntermedios>
<Nodo idNodo="2" />
<Nodo idNodo="3" />
<Nodo idNodo="4" />
<Nodo idNodo="5" />
</NodosIntermedios>
<NodosSumideros>
<Nodo idNodo="6" />
</NodosSumideros>
<Arcos>
<Arco idNodoInicio="1" idNodoFin="2" capacidad="16"/>
<Arco idNodoInicio="1" idNodoFin="3" capacidad="13"/>
<Arco idNodoInicio="2" idNodoFin="3" capacidad="10"/>
<Arco idNodoInicio="2" idNodoFin="4" capacidad="12"/>
<Arco idNodoInicio="3" idNodoFin="2" capacidad="4" />
<Arco idNodoInicio="3" idNodoFin="5" capacidad="14"/>
```

```
<Arco idNodoInicio="4" idNodoFin="3" capacidad="9"/>
<Arco idNodoInicio="4" idNodoFin="6" capacidad="20"/>
<Arco idNodoInicio="5" idNodoFin="4" capacidad="7"/>
<Arco idNodoInicio="5" idNodoFin="6" capacidad="4"/>
</Arcos>
</input>
```

Asimismo, los archivos XML de salida obedecen la estructura del siguiente DTD:

```
<!DOCTYPE output [
    <!ELEMENT input (FlujoMaximo | NodosFuentes | NodosIntermedios |
    NodosSumideros | Arcos)*>
        <!ELEMENT FlujoMaximo EMPTY>
            <!ATTLIST FlujoMaximo valor CDATA #REQUIRED>
        <!ELEMENT NodosFuentes (Nodo)>
            <!ELEMENT Nodo EMPTY>
                <!ATTLIST Nodo idNodo CDATA #REQUIRED>
        <!ELEMENT NodosIntermedios (Nodo)>
            <!ELEMENT Nodo EMPTY>
                <!ATTLIST Nodo idNodo CDATA #REQUIRED>
        <!ELEMENT NodosSumideros (Nodo)>
            <!ELEMENT Nodo EMPTY>
                <!ATTLIST Nodo idNodo CDATA #REQUIRED>
        <!ELEMENT Arcos (Arco)*>
            <!ELEMENT Arco EMPTY>
                <!ATTLIST Arco idNodoInicio CDATA #REQUIRED>
                <!ATTLIST Arco idNodoFin CDATA #REQUIRED>
                <!ATTLIST Arco capacidad CDATA #REQUIRED>
                <!ATTLIST Arco flujo CDATA #REQUIRED>
]>
```

Donde:

Elemento "output": El elemento output es el elemento base que contiene el resto de elementos del archivo XML.
Elemento "FlujoMaximo": Elemento que contiene el valor del flujo máximo de la red.
Atributo "valor" del elemento "FlujoMaximo": Valor del flujo máximo de la red.
Elemento "NodosFuentes": Representa el conjunto de nodos fuente de la red.
Elemento "Nodo": Representa un nodo fuente.
Atributo "idNodo" del elemento "Nodo": Valor del código del nodo fuente.
Elemento "NodosIntermedios": Representa el conjunto de nodos intermedios, es decir, los nodos que no son ni fuente ni sumidero, de la red.
Elemento "Nodo": Representa un nodo intermedio.
Atributo "idNodo" del elemento "Nodo": Valor del código del nodo intermedio
Elemento "NodosSumideros": Representa el conjunto de nodos sumideros de la red.
Elemento "Nodo": Representa un nodo sumidero.
Atributo "idNodo" del elemento "Nodo": Valor del código del nodo sumidero.
Elemento "Arcos": Representa el conjunto de arcos de la red
Elemento "Arco": Represente un arco de la red.
Atributo "idNodoInicio" del elemento "Arco": Valor del código del nodo de donde parte la red.
Atributo "idNodoFin" del elemento "Arco": Valor del código del nodo destino
Atributo "capacidad" del elemento "Arco": Valor de la capacidad del arco.
Atributo "flujo" del elemento "Arco": Valor del flujo que circula por el arco.

Así tenemos por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<output>
<FlujoMaximo valor="23" />
<NodosFuentes>
<Nodo idNodo="1" />
</NodosFuentes>
<NodosIntermedios>
<Nodo idNodo="2" />
<Nodo idNodo="3" />
<Nodo idNodo="4" />
```

<Nodo idNodo="5" />
</NodosIntermedios>
<NodosSumideros>
<Nodo idNodo="6" />
</NodosSumideros>
<Arcos>
<Arco idNodoInicio="1" idNodoFin="2" capacidad="16" flujo="16" />
<Arco idNodoInicio="1" idNodoFin="3" capacidad="13" flujo="7" />
<Arco idNodoInicio="2" idNodoFin="3" capacidad="10" flujo="8" />
<Arco idNodoInicio="2" idNodoFin="4" capacidad="12" flujo="12" />
<Arco idNodoInicio="3" idNodoFin="2" capacidad="4" flujo="4" />
<Arco idNodoInicio="3" idNodoFin="5" capacidad="14" flujo="11" />
<Arco idNodoInicio="4" idNodoFin="3" capacidad="9" flujo="0" />
<Arco idNodoInicio="4" idNodoFin="6" capacidad="20" flujo="19" />
<Arco idNodoInicio="5" idNodoFin="4" capacidad="7" flujo="7" />
<Arco idNodoInicio="5" idNodoFin="6" capacidad="4" flujo="4" />
</Arcos>
</output>

Capítulo 4: Construcción

En el presente capítulo se describen las tecnologías y frameworks utilizados para la implementación del software, asimismo, se explica detalladamente la construcción del algoritmo y su programación en el software, y, finalmente, se presenta el abordaje de las pruebas.

4.1. Construcción

A continuación se detallan las principales características involucradas en la construcción del software

4.1.1. Tecnologías utilizadas

Seguidamente se listan las tecnologías utilizadas para la etapa de construcción.

4.1.1.1. Lenguaje de programación

Se utilizará el lenguaje de programación JAVA para la implementación del software. Esta elección se basa en las siguientes razones:

- Facilidad de uso: Java provee facilidades en la programación, tales como la no manipulación directa de la memoria (no uso de punteros), objetos existentes

para la programación gráfica (con la librería Swing), y las propias de la programación orientada a objetos: herencia y polimorfismo (28).

- Portabilidad: El software desarrollado en Java puede ejecutarse en cualquier máquina independiente del sistema operativo que esta posea, lo cual le otorga al usuario la libertad de usar la plataforma que él desee.
- Libre: Tanto la plataforma de desarrollo de Java (JDK) como la mayoría de sus principales IDEs son gratuitos, lo cual reduce enormemente los costos de producción del software
- Diversidad de herramientas que facilitan el desarrollo: Existen innumerables librerías desarrolladas por terceros y publicadas en Internet a partir de las cuales se puede elaborar software más potente y con una mayor cantidad de funcionalidades (29).

4.1.1.2. Plataforma

Se utilizará la plataforma Java2 Platform, Standard Edition, v 1.4.2.

4.1.1.3. IDE

Se utilizará el IDE NetBeans IDE 6.7.1, por ser uno de los más amigables y contener facilidades en el desarrollo y una completa guía de ayuda al usuario

4.1.2. Frameworks

Se utilizará un framework adicional para facilitar la implementación del software:

- **JDOM**:
  La librería JDOM permite el manejo rápido, sencillo y seguro de los archivos XML, los cuales son bastante utilizados para el manejo de la información del software (30).

4.2. Construcción del algoritmo

A continuación se detallan las características principales del algoritmo.

4.2.1. Estructuras de datos

A continuación se definen las estructuras de datos a utilizar para implementar los conceptos relacionados al problema del flujo máximo:

1. Nodo

Los nodos son implementados a través de objetos cuya clase es la definida en la sección 2.3 Análisis, con sus respectivos atributos.

2. Arco

El arco es un objeto que contiene dos nodos, uno de origen y otro de destino; además, presenta también como características su capacidad, el flujo actual que circula por él y la capacidad residual del mismo

3. Red

En esencia, la red es un conjunto de nodos y arcos, en ese sentido, para la implementación de la misma, se ha empleado la notación matricial. Según esta, una red que contiene M nodos, sería representada por un matriz MxM, donde cada elemento (i, j), siendo i<=M y j<=M, de la matriz representará al arco que tiene como origen al nodo “i” y destino al nodo “j”. Sin embargo, para el presente proyecto, se ha realizado una variación a dicha representación, esta consiste en agregar dos nodos ficticios a la red existente, los cuales actuarán como súper-fuente y súper-sumidero, a fin de que el software pueda resolver calcular el flujo máximo en redes que contengan más de una fuente y/o más de un sumidero. Asimismo, de acuerdo a lo señalado en el capítulo 1, se deben crear también los respectivos arcos ficticios de capacidad infinita que vayan de la súper-fuente a los nodos fuentes, y de los sumideros al súper-sumidero.

Así, las estructuras de datos a utilizar para la representación de la red son:

a) Lista de nodos, a fin de identificar a cada uno de ellos, así como la cantidad total los mismos (M).
b) Matriz (M+2)x(M+2), donde cada elemento (i, j), siendo 0<=i<M+2 y 0<=j<M+2, de la matriz representa al arco que tiene como origen al nodo “i” y destino al nodo “j”. Cabe resaltar que el nodo “0” corresponde al nodo súper-fuente y el nodo M+1 al súper-sumidero

Adicionalmente, al ser el caso de múltiples fuentes y sumideros un caso más general del problema del flujo máximo, los nodos y arcos ficticios siempre se crearán en todas

las ejecuciones del algoritmo, aún si en la red existe únicamente una sola fuente y un solo sumidero.

4.2.2. Pseudocódigo del algoritmo

A continuación se presenta el pseudocódigo del algoritmo de Ford Fulkerson utilizado:

4.2.2.1. Inicialización de variables

En esta etapa se inicializan las estructuras de datos que se manejan en el algoritmo. Consiste en los siguientes procesos:

a) Creación de la lista de nodos: Donde, además de la generación de los nodos existentes de la red, también se crean los nodos súper-fuente y súper-sumidero.
b) Creación de la red: Se crean los arcos de la red, tanto los existentes como los ficticios que conectan la súper-fuente con las fuentes reales y el súper-sumidero con los sumideros existentes. Además, se genera la matriz (M+2)x(M+2) que contiene todos los arcos.
c) Creación de los arcos inversos: El algoritmo, para su ejecución, requiere la existencia de todos los arcos inversos a los arcos existentes de la red, es decir, los arcos que tengan como nodo origen el destino de otro arco, y como destino, el origen de dicho otro arco. Matemáticamente: dado un arco (u, v), su inverso será el arco (v, u). Si un arco pertenece a la red, pero su inverso no, este último debe ser creado de todas maneras, pero con una capacidad de cero, para asegurarse que no circule flujo por él.

4.2.2.2. Inicialización de flujos

Los flujos se inicializan en cero (0) para todos los arcos, y su capacidad residual en el valor de la capacidad de cada arco.

4.2.2.3. Búsqueda de trayectorias

Para la búsqueda de rutas, se definirá el concepto de nodo tope como el último nodo al cual se ha llegado con la búsqueda. Asimismo, se creará el atributo “pasó” en todos los nodos, el cual indicará si ya se ha incluido el nodo en la búsqueda de la trayectoria o no. Así, la búsqueda de trayectorias se realizará de la siguiente manera:

a) Se inicializa el atributo “pasó” de todos los nodos como falso

b) El primer tope será el nodo súper-fuente, este se agregará a la trayectoria y se colocará su indicador “pasó” en verdadero.
c) Se obtendrá un nodo que cumpla las siguientes características:
    - Sea destino de un arco que tenga como origen al tope.
    - Tenga el indicador “pasó” en falso.
    - El arco que los conecta tiene capacidad residual mayor que cero (puede aceptar más flujo).
d) Si existe el nodo del paso c), este es el nuevo tope, es agregado a la ruta y se coloca su indicador “pasó” en verdadero
e) De no existir el nodo, el tope actual “t” es retirado de la ruta buscada y se asigna como tope al último nodo agregado a la ruta antes de “t”.
f) Se repiten los procesos b) a e) hasta que el tope sea el súper-sumidero, o cuando el tope sea la súper-fuente y no se puedan agregar más nodos a la ruta.

4.2.2.4. Algoritmo

Dados:
G: Red (matriz de arcos)
Gf: Red residual de G (red formada por los arcos residuales de G)
s: nodo fuente
t: nodo sumidero
u,v: nodos de la red G
f(u,v): Flujo que circula por el arco (u,v)
p: trayectoria en la red
cr(p): capacidad residual de la trayectoria (p)
cr(u,v): capacidad residual del arco (u,v)
c(u,v): capacidad del arco (u,v)
FM: Flujo máximo de la red
L: lista de nodos

```
Ford-Fulkerson(G,s,t) {
        FM=0;
        inicializarVariables()
        inicializarFlujos()
```

```
Mientras (exista una trayectoria p desde s a t en la red residual Gf de G) {
        cf(p) = mín{cf(u,v) / (u,v) está en p};
        FM = FM+cf(p)
        Para cada arco (u,v) en p {
            cr(u,v) = cr(u,v) - cf(p)
            cr(v,u) = cr(v,u) + cf(p)
            Si c(u,v)>0 entonces {
                Si (f(u,v)+cr(p)<=c(u,v)) entonces {
                    f(u,v) = f(u,v) + cr(p)
                }
                Caso contrario {
                    exceso = f(u,v) + cr(p) – c(u,v)
                    f(u,v) = c(u,v)
                    f(v,u) = f(v,u) - exceso
                }
            }
            Caso contrario {
                f(v,u) = f(v,u) – cr(p)
            }
        }
    }
}
```

4.3. Pruebas

El proceso de pruebas permite asegurar la calidad y la no existencia de defectos en el sistema. El plan de pruebas permite administrar el proceso, las estrategias y asigna los roles primordiales en la ejecución de pruebas.

4.3.1. Abordaje de las pruebas

Se investiga los documentos de requerimientos del proyecto para diseñar las especificaciones de diseños de las pruebas, casos y procedimientos específicos.

4.3.1.1. Pruebas unitarias

Las pruebas unitarias consisten en realizar las pruebas sobre cada módulo diseñado de manera independiente, para identificar errores que cada uno de estos presente.

4.3.1.2. Pruebas integrales

Las pruebas integrales se ejecutan viendo los servicios como un todo y reportar fallas de integración del software. El objetivo principal de las pruebas de integración es de verificar que se cumple con todos los requerimientos especificados de dicho software.

4.3.1.3. Pruebas de desempeño y estrés

Las pruebas de desempeño miden que el sistema cumpla con las normas establecidas de desempeño definidas para él.

Las pruebas de estrés al software se realizan de manera que ocupa recursos en cantidades anormales. El objetivo de esta prueba es medir el comportamiento del sistema en situaciones extremas.

4.3.1.4. Criterios de aceptación o rechazo

Los criterios de aceptación o rechazo son las funcionalidades especificadas en la lista de requisitos del software.

4.3.1.5. Criterios de suspensión y renovación

Los criterios de suspensión y renovación proveen guías para diversas situaciones que se presentan en la ejecución de las pruebas que ameriten suspenderlas o reiniciarlas.

1. Criterios de suspensión

Se debe suspender cuando:

- El equipo computacional destinado para la prueba se encuentre inhabilitado o mal configurado para realizar las pruebas.
- El equipo computacional no cuenta con todos los requerimientos se software ni hardware necesarios para realizar las pruebas.

2. Criterios de renovación

Se debe renovar cuando

- Se obtenga un nuevo equipo que se encuentre en buen estado y cumpla los requerimientos de software y hardware necesarios

4.3.1.6. Entregables de las pruebas

1. Documentación de las pruebas

- Plan de pruebas (punto 4.3.)
- Catálogo de pruebas (donde se enumere cada uno de los casos de prueba) (ver Anexo G y Anexo H)
- Bitácora de pruebas (ver Anexo G y Anexo H)

2. Datos de las pruebas

- Precondición de la prueba en el catálogo de pruebas.
- Resultados esperados de la prueba en el catálogo de pruebas.
- Resultados obtenidos de la prueba en la bitácora de pruebas.

4.3.1.7. Tareas de las pruebas

Nro. Tarea	Tarea	Tareas predecesoras
1	Preparar los documentos de requerimientos del software	Ninguna.
2	Preparar las especificaciones de casos de prueba para el software.	Ninguna.
3	Ejecutar las pruebas de unitarias.	Tarea 2.
4	Ejecutar las pruebas de integración.	Tarea 3.
5	Solucionar los problemas surgidos al momento de ejecutar las pruebas.	Tarea 4.
6	Ejecutar las pruebas de regresión hasta que las	Tarea 5.

	pruebas hayan cumplido todos los criterios de aprobación	
7	Elaborar la bitácora de pruebas con los resultados de las pruebas	Tarea 6.

4.3.1.8. Necesidades mínimas del entorno de pruebas

- Hardware: Intel Pentium Core 2 Dúo2.5 GHz
- Sistema Operativo: Windows XP, Linux o Mac OS X
- Software: Java Runtime Environment 6

Capítulo 5: Observaciones, conclusiones y recomendaciones

En este capítulo se presentan las observaciones, conclusiones y recomendaciones generadas del desarrollo del presente proyecto.

5.1. Observaciones

La investigación de operaciones es una de las ramas más importantes de la matemática, pues busca solucionar problemas a los que se enfrentan hoy en día diversas instituciones a nivel mundial. Uno de dichos problemas es el del flujo máximo el cual consiste en calcular la mayor cantidad de material que puede circular a través de una red de nodos y arcos, cada arco con su respectiva capacidad máxima, de manera tal que en ningún caso la cantidad de material que fluye por cada arco supere la capacidad máxima del mismo.

Diferentes situaciones de la vida real pueden modelarse con el uso del problema del flujo máximo, entre ellas tenemos:

- La determinación del máximo número de conexiones que se pueden hacer en una red de comunicación
- El cálculo de la mayor cantidad de emparejamientos que se puede obtener, dados los gustos de cada individuo.

- El cálculo de la máxima cantidad de unidades que pueden transportarse desde los centros de producción a los centros de demanda. O su variante, cuando existe una cantidad mínima de productos que deben ir hacia ciertos puntos de demanda
- El cálculo de la máxima cantidad de vuelos que se pueden concretar entre dos ciudades entre las cuales existen puntos de escala, con limitaciones en cuanto a los aviones que pueden aterrizar dichas escalas.
- La selección óptima de proyectos que maximicen la relación beneficio/costo.

Si bien las situaciones anteriores son completamente disímiles entre sí, tienen en común que pueden tomar la forma de un problema del flujo máximo, por lo cual, la solución a las mismas es exactamente la misma, de ahí la importancia de este modelo matemático.

El algoritmo de Ford-Fulkerson es el principal método de solución al problema del flujo máximo (10). Básicamente, es un algoritmo iterativo que, en cada repetición, busca encontrar trayectorias en el grafo por donde aún pueda circular flujo, e ir incrementando el valor del flujo máximo en el valor de la mínima capacidad de entre todos los arcos que formen parte de la trayectoria encontrada, hasta que ya no se encuentren trayectorias por donde pueda circular más flujo. Mayor detalle puede encontrarse en el punto “4.2. Construcción del algoritmo”

5.2. Conclusiones

El software implementado puede ser utilizado tanto en la enseñanza de la investigación de operaciones como en aplicaciones prácticas como las señaladas en el punto 1.2.4 del capítulo 1 del presente documento.

Dada la cantidad de información e iteraciones que se utilizan en el problema del flujo máximo, y en los problemas de investigación de operaciones en general, se hace casi imposible la solución de dichos problemas manualmente, por lo cual se hace necesario que estos sean resueltos a través de programas computacionales.

Asimismo, dada la cantidad de datos que puede contener una red, ya sea para el problema del flujo máximo o para otros problemas de la investigación de operaciones, podría darse el caso que los métodos que calculen la solución exacta no encuentren

resultado por la cantidad de recursos que requerirían, por lo que podría ser necesario que se requieran nuevos algoritmos que, si bien no obtengan la solución exacta, encuentren soluciones aproximadas suficientemente adecuadas.

5.3. Recomendaciones y trabajos futuros

A futuro se podría personalizar el software para implementar alguna de las aplicaciones particulares que se modelen a través del problema del flujo máximo.

Asimismo se podrían implementar los otros dos algoritmos de solución a fin de compararlos y determinar los casos en los cuales cada uno de ellos es el más adecuado.

Adicionalmente, se podrían investigar algoritmos de inteligencia artificial que resuelven dicho problema con una mayor sofisticación.

Finalmente, se podrían implementar maneras alternativas de ingresar y visualizar las redes, como por ejemplo matrices o a través de interfaces gráficas.

Bibliografía

1. **Winston, Wayne L.** *Investigación de operaciones : aplicaciones y algoritmos.* 4ta. México D.F : Thompson Learning, 2005.

2. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.* 6ta. México D.F. : Mc Graw Hill, 2005.

3. **Flores, Luis A.** *Proyectos de tesis en Ingeniería Informática.* Lima, Lima, Perú : s.n., 15 de agosto de 2009.

4. —. *Guía para elaboración del Documento de Tesis.* Lima, Lima, Perú : s.n., 27 de marzo de 2009.

5. —. *Índices Base para Proyectos de Tesis en Ingeniería Informática.* Lima, Lima, Perú : s.n., 10 de agosto de 2009.

6. **Tolstoi, A. N.** *Methods of Finding Minimum Total Kilometrage in Cargo - Transportation Planning in Space.* 1930. págs. 23 - 55.

7. **Hudson, Ian.** *The Maximum Flow Problem.* Granville, Ohio, Estados Unidos : s.n., 1 de mayo de 2004.

8. **Hillier, Frederick S.** *Introducción a la investigación de operaciones.* 8va. México D.F. : McGraw-Hill, 2006. págs. 1-4.

9. **Taha, Hamdy A.** *Investigación de Operaciones.* 7. México D.F. : Prentice Hall, 2003. págs. 1-3.

10. **Cormen, Thomas H.** *Introduction to algorithms.* 2da. Boston : McGraw-Hill, 2001.

11. **Wayne, Kevin.** *Maximum Flow Applications.* Princeton, New Jersey, Estados Unidos : s.n., 2001.

12. **Larsen, Jesper y Clausen, Jens.** *The Max Flow Problem – Push-Relabel algorithms.*

13. **Goldberg, Andrew V.** *A new approach to the maximum flow problem.* Cambridge, Massachusetts, Estados Unidos : s.n., 1986.

14. **Rodríguez Villalobos, Alejandro.** Grafos. *Universidad Politécnica de Valencia.* [En línea] [Citado el: 14 de diciembre de 2009.] http://personales.upv.es/arodrigu/grafos/FordFulkerson.htm.

15. —. Alejandro Rodríguez Villalobos. *Universidad Politécnica de Valencia.* [En línea] [Citado el: 2009 de diciembre de 14.] http://personales.upv.es/arodrigu/Inicio.html.

16. —. ¿Qué es grafos? *Universidad Politécnica de Valencia.* [En línea] [Citado el: 14 de diciembre de 2009.] http://personales.upv.es/arodrigu/grafos/index.htm.

17. **Pacific Tech.** Graphing Calculator. *Pacific Tech.* [En línea] [Citado el: 16 de diciembre de 2009.] http://www.pacifict.com/.

18. **Ávila Herrera, Juan Félix.** Graficación animada. *Pacific Tech.* [En línea] [Citado el: 16 de diciembre de 2009.] http://www.pacifict.com/UsandoGraphingCalculator.pdf.

19. **Wolfram Research.** Wolfram Mathematica 7. *Wolfram Research.* [En línea] [Citado el: 16 de diciembre de 2009.] http://www.wolfram.com/products/mathematica/index.html.

20. —. History and background. *Wolfram Mathematica.* [En línea] [Citado el: 16 de diciembre de 2009.] http://www.wolfram.com/products/mathematica/history.html.

21. **Loubet, Beatriz.** *Ayuda Invop.* Mendoza, Argentina : s.n., 1998.

22. **Universidad de Valencia.** Universidad de Valencia. *Universidad de Valencia.* [En línea] [Citado el: 14 de abril de 2010.] http://www.uv.es/martinek/material/WinQSB2.0.pdf.

23. **Bello, Juan, y otros.** Universidad Santa María, Caracas - Venezuela. *Investigación de operaciones.* [En línea] Octubre de 2004. [Citado el: 21 de Abril de 2010.] http://www.investigacion-operaciones.com/Software/Manual%20WinQSB.pdf.

24. **Lindo Systems Inc.** Lindo Systems Products. *Sitio web de Lindo Systems.* [En línea] 15 de Enero de 2010. [Citado el: 2010 de Abril de 2010.] http://www.lindo.com/index.php?option=com_content&view=article&id=2&Itemid=10.

25. **J., Ponce, G., Solis y L., Ulfe.** Guía básica de LINGO. *Investigación de operaciones S.A.* [En línea] [Citado el: 21 de Abril de 2010.] http://www.iosa.com.pe/descargas/Programacion%20Lineal/Guia-LINGO%20(Intro%20y%20modelos).pdf.

26. **Canizo, Erica y Lucero, Paola.** Universidad Tecnológica Nacional. *Facultad Regional Mendoza.* [En línea] 2002. [Citado el: 2010 de Abril de 21.] http://www.frm.utn.edu.ar/ioperativa/lingo_lindo.pdf.

27*GraphXML - An XML-based graph description format*AmsterdanHolanda

28. **O. Pflucker, C. Zapata, J. Baldeón.** *Lenguajes de programación 2. Temas de estudio.* 3rd Edition. Lima : Pontificia Universida Católica del Perú, 2006. págs. 2-5.

29. **H.M. Deitel, P.J. Deitel.** *Java, how to program.* 5th Edition. s.l. : Pearson Education, 2003. págs. 1-18.

30. **JDOM Project.** jdom.org. *Documentation.* [En línea] [Citado el: 1 de January de 2012.] http://www.jdom.org/downloads/docs.html.